Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студент *Кравченко Данило Ігорович*
(ПІБ)

академічної групи *121-20-2*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка програмного забезпечення для ергономічного налаштування реєстру персональних комп'ютерів*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*проф. Лактіонов І.С.*			
розділів:				
спеціальний	*проф. Лактіонов І.С.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Мартиненко А.А.*			

Дніпро
2024

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студент *121-20-2* *Кравченко Данило Ігорович*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного забезпечення для ергономічного налаштування реєстру персональних комп'ютерів*

затверджена наказом ректора НТУ «ДП» від *23.05.2024 р.* № *469-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***10.06.2024 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***14.06.2024 р.***

Завдання видав _____ *проф. Лактіонов І.С*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Кравченко Д.І.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2024 р.*

Термін подання кваліфікаційної роботи до ЕК: *21.06.2024 р.*

РЕФЕРАТ

Пояснювальна записка: 84 с., 18 рис., 4 дод., 6 табл., 25 джерел.

Об'єкт розробки: програмне забезпечення для налаштування реєстру операційної системи на персональних комп'ютерах.

Мета кваліфікаційної роботи: створення зручного та ефективного інструменту для безпечного редагування системного реєстру Windows, що покращить продуктивність і зручність використання.

Актуальність роботи зумовлена зростаючою складністю операційних систем Windows та збільшенням кількості користувачів, які потребують ефективних інструментів для оптимізації роботи своїх комп'ютерів. В умовах постійного оновлення програмного забезпечення та зростання вимог до продуктивності, створення безпечного та зручного інструменту для редагування реєстру стає критично важливим. Це дозволить користувачам і системним адміністраторам ефективно керувати налаштуваннями системи, уникаючи потенційних ризиків та помилок, що можуть виникнути при ручному редагуванні реєстру.

У вступі розглядається важливість налаштування реєстру для коректної роботи Windows та додатків. Аналізуються проблеми та ризики некоректного редагування реєстру вручну. Наголошується на необхідності зручного та безпечного програмного забезпечення для редагування реєстру. Визначається мета роботи та її актуальність.

Перший розділ присвячений дослідженню предметної області та постановці завдання. Розглядається структура та функції реєстру Windows, його важливість, типові проблеми та потреби користувачів. Формулюються вимоги до функціональності, зручності, безпеки та сумісності програмного забезпечення.

У другому розділі проводиться огляд існуючих рішень для редагування реєстру Windows, їх переваги та недоліки. Обґрунтовується вибір платформи та технологій. Описуються етапи проектування, розробки та тестування програмного забезпечення, його архітектури, алгоритмів роботи, способів взаємодії з користувачем та заходів безпеки.

Економічний розділ містить розрахунки трудомісткості розробки, вартості робіт та часу для створення програмного забезпечення.

Практичне завдання полягає у розробці повноцінного програмного продукту для безпечного та зручного редагування реєстру Windows з урахуванням досліджень та аналізу існуючих рішень.

Ключові слова: РЕЄСТР WINDOWS, НАЛАШТУВАННЯ РЕЄСТРУ, РЕДАГУВАННЯ РЕЄСТРУ, ОПТИМІЗАЦІЯ ПРОДУКТИВНОСТІ, БЕЗПЕКА РЕЄСТРУ, ЗРУЧНІСТЬ КОРИСТУВАННЯ, C++, .NET, WPF, XAML.

ABSTRACT

Explanatory note: 84 p., 33 pic., 3 add., 25 sources.

Object of development: software for configuring the registry of the operating system on personal computers.

The goal of the qualification work: to create a convenient and effective tool for safe editing of the Windows system registry, which will improve performance and usability.

The relevance of this work is driven by the increasing complexity of Windows operating systems and the growing number of users who require effective tools to optimize their computers' performance. In the context of constant software updates and increasing performance requirements, creating a safe and user-friendly tool for registry editing becomes critically important. This will allow users and system administrators to efficiently manage system settings while avoiding potential risks and errors that can occur with manual registry editing.

The introduction discusses the importance of registry settings for the correct operation of Windows and applications. Problems and risks of incorrect manual editing of the registry are analyzed. Emphasizes the need for user-friendly and secure registry editing software. The purpose of the work and its relevance are determined.

The first chapter is dedicated to researching the subject area and setting the task. The structure and functions of the Windows registry, its importance, common problems and user needs are considered. Requirements for software functionality, convenience, security and compatibility are formulated.

The second section provides an overview of existing solutions for editing the Windows registry, their advantages and disadvantages. The choice of platform and technologies is justified. The stages of design, development and testing of software, its architecture, work algorithms, methods of interaction with the user and security measures are described.

The economic section contains calculations of development effort, cost of work and time for software creation.

The practical task is to develop a full-fledged software product for safe and convenient editing of the Windows registry, taking into account research and analysis of existing solutions.

Keywords: WINDOWS REGISTRY, REGISTRY CONFIGURATION, REGISTRY EDITING, PRODUCTIVITY OPTIMIZATION, REGISTRY SECURITY, USABILITY, C++, .NET, WPF, XAML.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СУБД – система управління базами даних

ПК – персональний комп’ютер

DBMS – database management system

ВСТУП

У сучасному світі персональні комп'ютери широко використовуються в різних сферах діяльності. Однак, під час їх експлуатації, часто виникають проблеми, пов'язані з налаштуванням та оптимізацією реєстру операційної системи. Реєстр є критично важливою частиною операційної системи Windows, яка зберігає усю інформацію про налаштування, параметри та інші важливі дані для коректної роботи комп'ютера та встановлених програм.

Неправильне налаштування реєстру може призвести до порушення роботи програм, системних збоїв та інших небажаних наслідків. Крім того, з часом реєстр може розрастатися та сповільнювати роботу комп'ютера. Тому актуальним є розробка програмного забезпечення, яке дозволить ергономічно налаштувати реєстр, оптимізувати його роботу та усунути потенційні проблеми.

Мета роботи: Розробити програмне забезпечення для ергономічного налаштування реєстру персональних комп'ютерів, яке дозволить користувачам легко та зручно керувати налаштуваннями реєстру, оптимізувати його роботу та усувати потенційні проблеми.

Об'єкт розробки: Процес управління та налаштування реєстру операційної системи Windows на персональних комп'ютерах.

Задачі:

1. Аналіз існуючих рішень та підходів до налаштування реєстру операційної системи Windows.
2. Визначення основних вимог до програмного забезпечення для ергономічного налаштування реєстру.
3. Розробка зручного та інтуїтивно зрозумілого інтерфейсу користувача.
4. Реалізація функціоналу для перегляду, пошуку, редагування та видалення ключів реєстру.
5. Впровадження механізмів оптимізації та очищення реєстру від зайвих або пошкоджених записів.

6. Забезпечення безпеки та цілісності даних реєстру під час налаштування.

7. Тестування та налагодження програмного забезпечення.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Загальні відомості з предметної галузі

У підрозділі "Загальні відомості з предметної галузі" розглядається сучасний стан проблеми налаштування реєстру операційної системи Windows [1, 6, 14]. Аналіз аналогів показує, що на ринку існують різноманітні програми для роботи з реєстром, проте більшість із них мають складний інтерфейс, обмежений функціонал або вимагають додаткових плат них послуг [4, 19].

Операційна система Windows використовує кілька різних видів реєстрів для зберігання налаштувань, конфігурацій та інших важливих даних, необхідних для коректного функціонування системи та встановлених програм [2, 17]. Кожен вид реєстру відповідає за певну сферу налаштувань і має свою унікальну структуру та призначення.

Системний реєстр (System Registry)

Системний реєстр є одним з найважливіших реєстрів в операційній системі Windows. Він містить критично важливі налаштування, пов'язані з ядром операційної системи, драйверами пристроїв, параметрами безпеки та іншими системними компонентами. Системний реєстр зберігається у файлах SYSTEM та SYSTEM.LOG у директорії %SystemRoot%\system32\config\ і завантажується під час запуску операційної системи.

Будь-які помилки або пошкодження в системному реєстрі можуть призвести до серйозних збоїв у роботі комп'ютера, тому редагування цього реєстру повинно здійснюватися лише досвідченими користувачами або адміністраторами системи. Системний реєстр відіграє ключову роль у забезпеченні стабільної та безпечної роботи операційної системи Windows [1].

Реєстр користувача (User Registry) (Рис. 1.1).

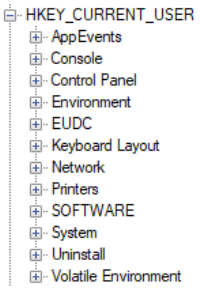

Рис. 1.1. Под ключи реєстра користувача

Реєстр користувача містить налаштування, пов'язані з окремим користувачем комп'ютера. Сюди входять параметри оболонки, розташування папок, налаштування програм, персональні налаштування та інші дані, специфічні для кожного користувача. Реєстр користувача зберігається у файлах DEFAULT та DEFAULT.LOG для нових користувачів, а також NTUSER.DAT та NTUSER.DAT.LOG для поточного користувача в директорії %UserProfile% (Рис. 1.2).

Рис. 1.2. Файл реєстра користувача

Завдяки реєстру користувача, кожен користувач може мати свій унікальний набір налаштувань та персоналізації робочого середовища операційної системи. Це забезпечує зручність та ергономічність використання комп'ютера для різних користувачів [2].

Реєстр програм (Software Registry) (Рис. 1.3).

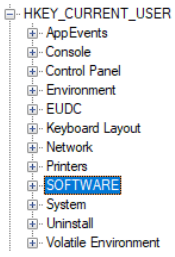

Рис. 1.3. Ключ реєстра програм

Реєстр програм зберігає інформацію про встановлені на комп'ютері програми, їх параметри, налаштування та інші дані, необхідні для коректної роботи програмного забезпечення. Цей реєстр розміщується у гілках HKEY_CURRENT_USER\SOFTWARE та HKEY_LOCAL_MACHINE\SOFTWARE (Рис. 1.4).

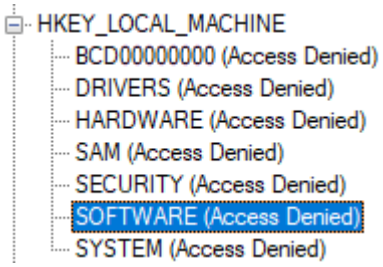

Рис. 1.4. Ключ локальной машини реєстра програм

Під час встановлення нової програми відповідні записи додаються до реєстру програм, що дозволяє операційній системі та іншим програмам правильно взаємодіяти з нею. Реєстр програм відіграє важливу роль у забезпеченні сумісності та стабільності роботи програмного забезпечення в операційній системі Windows [3].

Реєстр класів (Class Registry) (Рис. 1.5)

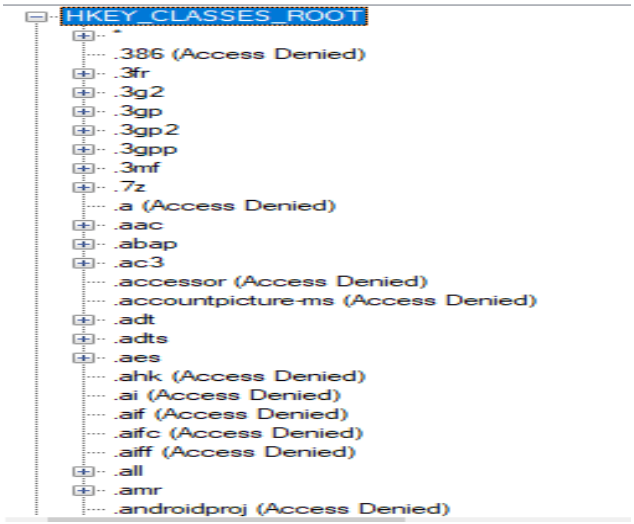

Рис. 1.5. Под ключи реєстра класів

Реєстр класів зберігає асоціації між різними типами файлів та програмами, призначеними для їх обробки. Він також містить інформацію про властивості файлів, їх значки та інші метадані. Реєстр класів розміщується в гілці HKEY_CLASSES_ROOT.

Завдяки реєстру класів, операційна система Windows може правильно відкривати файли відповідними програмами, відображати їх значки та надавати користувачеві додаткову інформацію про тип файлу. Це забезпечує зручність роботи з різноманітними типами файлів та сумісність між програмами [4].

Реєстр служб (Services Registry) (Рис. 1.6**).**

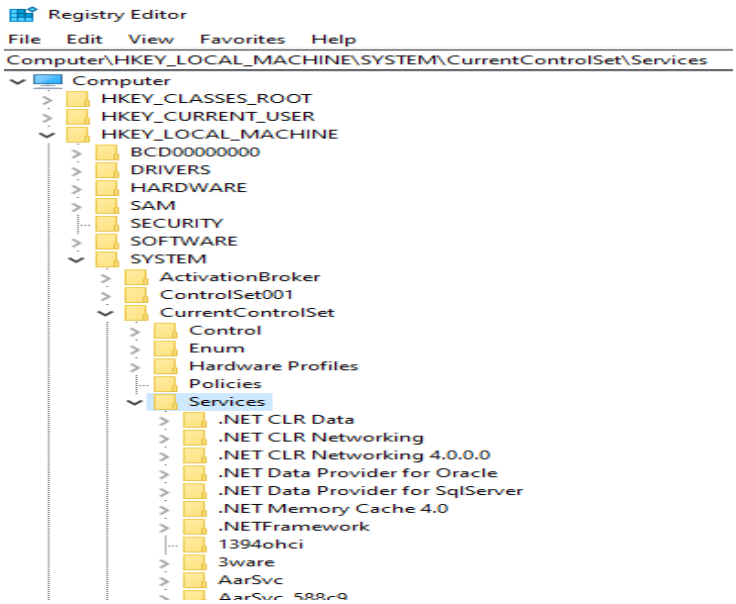

Рис. 1.6. Под ключи реєстра служб

Реєстр служб містить інформацію про системні служби Windows, їх налаштування та параметри запуску. Він знаходиться у гілці HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services.

Системні служби є важливими компонентами операційної системи, які відповідають за різноманітні функції, такі як антивірусний захист, мережеві з'єднання, планувальник завдань та багато іншого. Реєстр служб дозволяє керувати цими службами, налаштовувати їх параметри та забезпечувати коректну роботу системи [5].

Реєстр профілів обладнання (Hardware Profiles Registry) (Рис. 1.7**).**

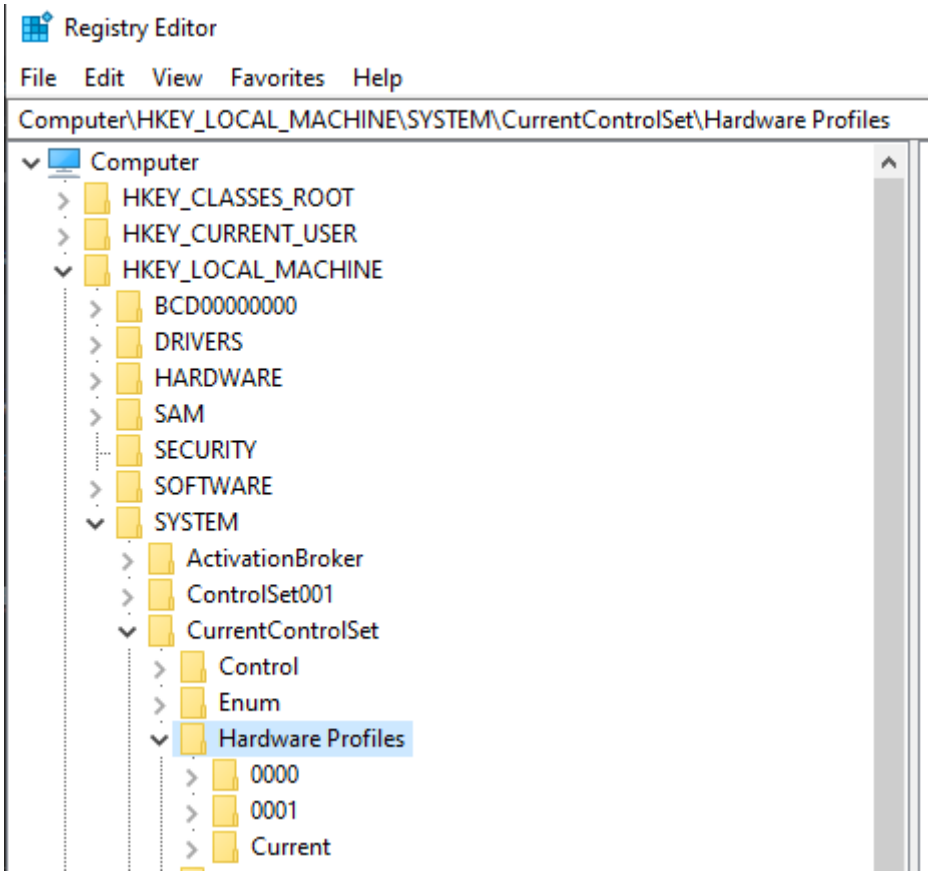

Рис. 1.7. Под ключи реєстра профілів обладнання

Реєстр профілів обладнання зберігає налаштування та конфігурації різних апаратних профілів для комп'ютера. Він знаходиться у гілці HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Hardware Profiles.

Апаратні профілі використовуються для налаштування комп'ютера під різні сценарії використання, наприклад, роботу в офісі, вдома або в мобільному режимі. Реєстр профілів обладнання дозволяє зберігати та завантажувати відповідні налаштування для кожного профілю, забезпечуючи оптимальну конфігурацію системи для різних умов [6].

Після аналізу всіх реєстрів операційної системи Windows за визначеними критеріями, найбільш оптимальним вибором для розробки програмного забезпечення для ергономічного налаштування реєстру є реєстр програм (Software Registry). Він задовольняє всі критерії: доступний для звичайних користувачів, містить основну інформацію про налаштування встановлених програм, його редагування не створює критичних ризиків для стабільності

системи, і він дозволяє персоналізувати налаштування для окремих користувачів.

Тому, реєстр програм буде використовуватися в якості основного об'єкта для розробки програмного забезпечення для ергономічного налаштування реєстру персональних комп'ютерів.

1.2. Призначення розробки та галузь застосування

Повна назва розробленого програмного забезпечення: "ErgonReg - Ергономічний налаштувач реєстру Windows".

Основні терміни та визначення:

- **Реєстр Windows (Windows Registry)** - централізована ієрархічна база даних, що містить критично важливу інформацію про налаштування, параметри, опції та інші дані операційної системи Windows, встановлених програм та користувачів.

- **Ергономіка (Ergonomics)** - мультидисциплінарна галузь, що вивчає взаємодію людини з технічними системами та навколишнім середовищем, з метою оптимізації продуктивності, безпеки, комфорту та задоволеності користувачів.

- **Налаштування реєстру (Registry Configuration)** - процес модифікації ключів та значень реєстру для зміни поведінки, функціональності та налаштувань операційної системи, програмного забезпечення або апаратних компонентів.

- **Оптимізація реєстру (Registry Optimization)** - комплекс заходів щодо очищення реєстру від зайвих, застарілих або пошкоджених записів, дефрагментації та оптимізації структури реєстру для підвищення продуктивності та стабільності системи.

Необхідність розробки програмного забезпечення "ErgonReg" обумовлена такими чинниками:

1. Складність та високі ризики, пов'язані з ручним редагуванням реєстру за допомогою вбудованих інструментів Windows, особливо для користувачів з обмеженим досвідом роботи.

2. Відсутність зручного, інтуїтивного та ергономічного інтерфейсу для налаштування, оптимізації та очищення реєстру, придатного для використання широким колом користувачів різного рівня досвіду.

3. Потреба в централізованому, надійному та безпечному інструменті для комплексного керування реєстром, що забезпечить ефективне налаштування, оптимізацію та усунення потенційних проблем, пов'язаних з реєстром.

Галузі застосування розробленого програмного забезпечення "ErgonReg":

1. Персональні комп'ютери з операційною системою Windows для домашнього та офісного використання, де програма дозволить користувачам самостійно налаштовувати та оптимізувати роботу їх комп'ютерів.

2. Технічна підтримка та обслуговування комп'ютерів у малих і середніх організаціях, де програма стане зручним інструментом для IT-фахівців та адміністраторів для ефективного керування налаштуваннями реєстру.

3. Навчальні заклади, де програма може використовуватися для викладання основ роботи з операційними системами, налаштування реєстрів та ознайомлення студентів з принципами ергономіки та оптимізації продуктивності комп'ютерів.

4. Ремонтні майстерні та сервісні центри, де програма дозволить технічним фахівцям швидко та безпечно оптимізувати роботу комп'ютерів клієнтів, усуваючи проблеми, пов'язані з реєстром.

Таким чином, "ErgonReg" призначений для надання користувачам зручного, безпечного та ергономічного інструменту для керування реєстром Windows, що дозволить їм ефективно налаштовувати операційну систему та встановлене програмне забезпечення, оптимізувати роботу комп'ютера та усувати потенційні проблеми, пов'язані з реєстром, незалежно від їх рівня технічних знань та досвіду.

1.3. Підстави для розробки

В кінці навчання, студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником проекту, випускаючою кафедрою.

Підставою для розробки кваліфікаційної роботи на тему «Розробка програмного забезпечення для ергономічного налаштування реєстру персональних комп'ютерів» є наказ № 469-с від 23.05.2024 р. по Національному технічному університету «Дніпровська політехніка».

1.4. Підрозділ «Постановка завдання»

Мета розробки: Створення ефективного та ергономічного програмного забезпечення для комплексного управління реєстром операційної системи Windows на персональних комп'ютерах, що забезпечить оптимізацію роботи комп'ютера, підвищену продуктивність та стабільність функціонування системи.

Призначення: Надати користувачам персональних комп'ютерів з операційною системою Windows зручний та безпечний інструмент для налаштування, моніторингу, оптимізації та очищення реєстру, який дозволить підвищити ефективність роботи комп'ютера та запобігти потенційним проблемам, пов'язаним з некоректним функціонуванням реєстру.

Обґрунтування доцільності: Розроблене програмне забезпечення дозволить усунути недоліки існуючих рішень для керування реєстром Windows, такі як складний інтерфейс, обмежений функціонал або потенційні ризики для стабільності системи. Наявність зручного та надійного інструменту для ергономічного налаштування реєстру сприятиме підвищенню продуктивності роботи користувачів з комп'ютером, зменшенню ризиків нестабільної роботи та усуненню проблем, пов'язаних з некоректною конфігурацією реєстру.

Об'єкт управління: Реєстр операційної системи Windows на персональних комп'ютерах, що включає системний, користувацький, програмний та інші реєстри.

Опис завдання: Забезпечити ефективне централізоване керування реєстром персональних комп'ютерів з операційною системою Windows, що включає моніторинг стану реєстру, його налаштування, оптимізацію та очищення від застарілих або пошкоджених даних, з метою оптимізації роботи комп'ютера та запобігання потенційним проблемам, пов'язаним з некоректною конфігурацією реєстру.

Зміст підрозділу:

1. Обґрунтування мети та завдань розробки
2. Опис об'єкта інформаційної системи та його структури
3. Вимоги до організації збору, обробки та передачі вхідної інформації
4. Опис призначення вихідної інформації
5. Умови припинення вирішення завдання автоматизованим способом
6. Зв'язок розробленої системи з іншими задачами
7. Розподіл функцій між користувачами та технічними засобами

1. Мета та завдання розробки:

- Створення інтуїтивного та ергономічного інтерфейсу користувача для зручного доступу до реєстру Windows.

- Забезпечення можливості перегляду, пошуку, редагування та видалення ключів і значень реєстру.

- Реалізація функціоналу для моніторингу стану реєстру, виявлення потенційних проблем та помилок.

- Впровадження механізмів оптимізації та очищення реєстру від зайвих, застарілих або пошкоджених записів.

- Забезпечення безпеки та цілісності даних реєстру під час виконання операцій налаштування та оптимізації.

- Інтеграція з операційною системою Windows для ефективної взаємодії з реєстром.

2. Об'єкт інформаційної системи та його структура:

- Реєстр операційної системи Windows, що складається з окремих реєстрів (системний, користувацький, програмний тощо).

- Ієрархічна структура реєстрів, що містить ключі, підключі та відповідні значення.

- Параметри та налаштування системних компонентів [3, 5, 7, 9, 12, 13], програм та користувачів, які зберігаються в реєстрі.

3. Вимоги до організації збору, обробки та передачі вхідної інформації:

- Забезпечити безпечний доступ до даних реєстру для їх зчитування, аналізу та обробки.

- Реалізувати механізми контролю цілісності та несуперечності даних реєстру.

- Організувати ефективний збір та аналіз інформації про стан реєстру, виявлення потенційних проблем та помилок.

- Забезпечити можливість передачі даних реєстру для їх оптимізації та очищення.

4. Призначення вихідної інформації:

- Надання користувачеві зрозумілої інформації про поточний стан реєстру, виявлені проблеми та помилки.

- Відображення результатів операцій налаштування, оптимізації та очищення реєстру.

- Формування звітів про виконані дії та внесені зміни в реєстр.

5. Умови припинення вирішення завдання автоматизованим способом:

- Виникнення критичних помилок або порушення цілісності даних системного реєстру.

- Втрата зв'язку з ядром операційної системи або компонентами реєстру.

- Виявлення загрози безпеці або цілісності системи під час виконання операцій з реєстром.

6. Зв'язок розробленої системи з іншими задачами:

- Інтеграція з операційною системою Windows [25] для ефективної взаємодії з реєстром.

- Можливість взаємодії з іншими програмами для управління системними ресурсами та налаштуваннями.

- Використання стандартних інтерфейсів та протоколів для обміну даними з іншими компонентами системи.

7. Розподіл функцій між користувачами та технічними засобами:

- Користувачі: ініціювання операцій з реєстром, налаштування параметрів, перегляд та редагування ключів і значень, вибір режимів оптимізації та очищення.

- Технічні засоби: забезпечення інтерфейсу користувача, доступу до реєстру, виконання операцій з реєстром, аналіз та обробка даних, забезпечення безпеки та цілісності системи.

Таким чином, даний підрозділ охоплює всі ключові аспекти постановки завдання для розробки програмного забезпечення для ергономічного налаштування реєстру операційної системи Windows, включаючи мету, призначення, опис об'єкта управління, вимоги до організації обробки даних, умови припинення вирішення завдання, зв'язок з іншими системними задачами та розподіл функцій між користувачами та технічними засобами.

1.5. Вимоги до програмного забезпечення "ErgonReg" для ергономічного налаштування реєстру

Розробка програмного забезпечення для ергономічного налаштування реєстру персональних комп'ютерів є актуальною задачею, оскільки вона допомагає забезпечити зручність та ефективність роботи користувачів з комп'ютерами [8, 10, 16]. Ергономічне налаштування реєстру персональних комп'ютерів полягає у налаштуванні системних параметрів та конфігурацій для створення оптимального робочого середовища, що відповідає індивідуальним потребам та особливостям кожного користувача [18, 20].

1.5.1. Вимоги до функціональних характеристик

Програма "ErgonReg" повинна надавати користувачам можливість гнучкого налаштування ключових параметрів реєстру, таких як розміри

шрифтів, колірні схеми, розташування елементів інтерфейсу, швидкість реакції системи та інші. Цей процес має відбуватися через інтуїтивно зрозумілий графічний інтерфейс, що дозволяє користувачам легко знаходити та змінювати потрібні налаштування.

Програмного забезпечення повинна забезпечувати можливість створення та збереження профілів налаштувань для різних користувачів або робочих сценаріїв. Це дасть змогу швидко переключатися між різними конфігураціями залежно від потреб.

Крім того, "ErgonReg" має включати функції автоматичного налаштування на основі аналізу особливостей користувача, таких як вік, зір, вподобання тощо, щоб забезпечити максимально комфортний робочий процес.

1.5.2. Вимоги до інформаційної безпеки

Програма "ErgonReg" повинна гарантувати захист конфіденційних даних користувачів, таких як особисті налаштування та профілі. Доступ до цієї інформації має бути захищений паролем або іншими методами автентифікації.

Необхідно забезпечити цілісність даних, щоб унеможливити випадкове або навмисне пошкодження налаштувань реєстру. Програма повинна регулярно створювати резервні копії налаштувань для відновлення у випадку збою або помилки.

Також важливо передбачити захист від несанкціонованого доступу до програмного забезпечення та його компонентів, щоб запобігти можливому зловмисному втручанню в роботу системи.

1.5.3. Вимоги до складу та параметрів технічних засобів

Програма "ErgonReg" повинна бути сумісною з різними операційними системами та апаратними конфігураціями персональних комп'ютерів. Необхідно визначити мінімальні системні вимоги, такі як обсяг оперативної пам'яті,

швидкість процесора та вільне місце на жорсткому диску, для забезпечення належного функціонування програми.

Крім того, програма має підтримувати різні типи периферійних пристроїв, таких як монітори, клавіатури, миші та інші пристрої введення/виведення, для забезпечення максимальної зручності роботи користувачів.

1.5.4. Вимоги до інформаційної та програмної сумісності

Програмне забезпечення "ErgonReg" повинно бути розроблене з використанням сучасних мов програмування та технологій, що забезпечують сумісність з іншими програмними продуктами та системами.

Необхідно дотримуватися стандартів та протоколів обміну даними, щоб забезпечити безперебійну інтеграцію з іншими додатками та системами, які можуть використовуватися на персональних комп'ютерах користувачів.

"ErgonReg" повинна підтримувати імпорт та експорт налаштувань у стандартних форматах файлів, що дозволить обмінюватися конфігураціями між різними користувачами або переносити їх на інші комп'ютери.

Загалом, розробка програмного забезпечення для ергономічного налаштування реєстру персональних комп'ютерів є складним завданням, що вимагає ретельного планування, аналізу вимог користувачів та дотримання стандартів безпеки та сумісності. Лише за умови врахування всіх цих аспектів можна створити якісний продукт, який забезпечить зручну та ефективну роботу користувачів з персональними комп'ютерами.

Висновок

У першому розділі кваліфікаційної роботи проведено детальний аналіз предметної галузі та сформульовано основні завдання розробки програмного забезпечення "ErgonReg" для ергономічного налаштування реєстру операційної системи Windows.

Розглянуто сучасний стан проблеми налаштування реєстру Windows та аналіз існуючих аналогів. Визначено, що багато програм мають складний інтерфейс або обмежений функціонал, що створює потребу в розробці більш зручного та ефективного рішення.

Описано структуру різних видів реєстрів Windows, таких як системний, користувацький, програмний, класів, служб та профілів обладнання. Визначено їх важливість для коректного функціонування системи та програм, а також ризики, пов'язані з їх редагуванням.

Обґрунтовано вибір реєстру програм (Software Registry) як основного об'єкта для розробки програмного забезпечення, оскільки він задовольняє критеріям доступності, безпеки та персоналізації налаштувань.

Розроблено концепцію програмного забезпечення "ErgonReg" та визначено його призначення для широкого кола користувачів, включаючи домашніх користувачів, IT-фахівців, навчальні заклади та сервісні центри.

Поставлено завдання створення інтуїтивного та ергономічного інтерфейсу користувача, забезпечення безпеки даних, сумісності з різними операційними системами та пристроями, а також підтримки сучасних стандартів програмування та обміну даними.

Таким чином, перший розділ охоплює всі ключові аспекти аналізу предметної галузі та постановки завдання для розробки програмного забезпечення, що дозволяє перейти до безпосереднього проектування та розробки продукту у наступних розділах роботи.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Функціональне призначення:

Програма "ErgonReg" призначена для забезпечення зручного та безпечного інструменту для керування реєстром операційної системи Windows. Основні функціональні можливості програми включають:

- Відображення структури реєстру у вигляді дерева (TreeView): Це дозволяє користувачам легко навігувати між різними гілками реєстру, бачити їх взаємозв'язок та швидко знаходити потрібні ключі та значення.

- Зручний інтерфейс для редагування, видалення та створення ключів і значень реєстру: Користувачі можуть легко додавати нові ключі та значення, змінювати існуючі або видаляти непотрібні, використовуючи інтуїтивно зрозумілий графічний інтерфейс.

- Автоматичне резервне копіювання реєстру перед внесенням змін: Програма автоматично створює резервну копію поточного стану реєстру перед тим, як користувач вносить будь-які зміни, що забезпечує можливість швидкого відновлення у разі помилки.

- Виявлення та виправлення помилок у реєстрі: Програма може автоматично сканувати реєстр на наявність помилок або некоректних записів та пропонувати користувачу варіанти їх виправлення.

- Оптимізація реєстру для підвищення продуктивності системи: Програма може аналізувати структуру реєстру, видаляти зайві та застарілі записи, а також дефрагментувати реєстр, що покращує швидкість доступу до даних та загальну продуктивність системи.

Експлуатаційне призначення:

Програма "ErgonReg" надає користувачам можливість легко та безпечно керувати реєстром, що дозволяє:

- Зменшити ризик виникнення системних помилок: Автоматизоване резервне копіювання та виявлення помилок допомагають запобігти критичним збоєм у роботі системи, які можуть виникнути через некоректні зміни в реєстрі.

- Підвищити продуктивність та стабільність роботи операційної системи: Оптимізація реєстру дозволяє видаляти непотрібні та пошкоджені записи, що позитивно впливає на швидкість та стабільність роботи системи.

- Автоматизувати процеси резервного копіювання та відновлення реєстру: Це забезпечує додатковий рівень безпеки, оскільки користувачі можуть швидко відновити реєстр до попереднього стану у разі виникнення проблем.

- Забезпечити зручний інтерфейс для адміністраторів систем та звичайних користувачів: Програма пропонує простий у використанні інтерфейс, що робить процес управління реєстром доступним навіть для користувачів без спеціальних технічних знань, одночасно надаючи достатній функціонал для досвідчених IT-адміністраторів.

2.2. Опис застосованих математичних методів

У розробці програмного забезпечення "ErgonReg" застосовуються наступні математичні методи та алгоритми:

1. Алгоритми хешування: Для ефективного пошуку та виявлення дублікатів записів у реєстрі використовуються хеш-функції, зокрема SHA-256. Це дозволяє швидко порівнювати великі обсяги даних.

2. Статистичні методи: Для аналізу стану реєстру та виявлення аномалій застосовуються базові статистичні методи, такі як:

- Розрахунок середніх значень та медіан для числових параметрів реєстру
- Визначення стандартного відхилення для виявлення значень, що сильно відрізняються від норми
- Використання методу z-score для нормалізації даних та виявлення викидів

3. Алгоритми сортування: Для оптимізації роботи з даними реєстру використовуються ефективні алгоритми сортування, такі як QuickSort, що дозволяє швидко впорядковувати великі обсяги даних за різними критеріями.

4. Методи стиснення даних: Для ефективного зберігання резервних копій реєстру застосовуються алгоритми стиснення без втрат, наприклад, алгоритм Лемпеля-Зіва-Велча (LZW).

Ці методи реалізовані з використанням стандартних бібліотек .NET Framework та оптимізовані для роботи з великими обсягами даних реєстру Windows.

2.3. Опис використаної архітектури та шаблонів проектування

Архітектура (Рис 1.8):

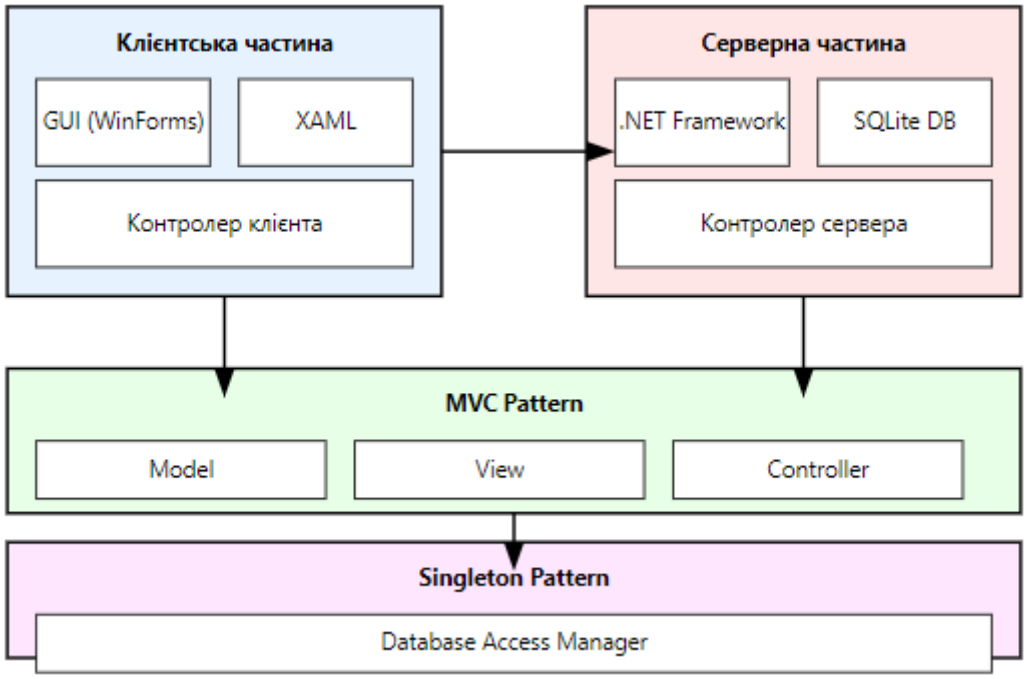

Рис 1.8. Схема ілюструє основні компоненти.

Програмне забезпечення "ErgonReg" розроблене з використанням клієнт-серверної архітектури [24], яка забезпечує ефективний розподіл обов'язків між клієнтською та серверною частинами програми.

- Клієнтська частина: Клієнтська частина відповідає за взаємодію з користувачем та надання інструментів для перегляду, редагування, видалення та створення ключів і значень реєстру. Вона реалізована за допомогою технологій WinForms і XAML, що забезпечує зручний графічний інтерфейс користувача (GUI). Користувачі можуть виконувати всі операції з реєстром безпосередньо через цей інтерфейс.

- Серверна частина: Серверна частина зберігає базу даних з резервними копіями реєстру та обробляє запити клієнта. Вона реалізована з використанням .NET Framework та бази даних [21, 23] SQLite, що дозволяє ефективно керувати резервними копіями та забезпечувати швидке відновлення у разі необхідності. Сервер обробляє запити на створення резервних копій, відновлення реєстру та інші операції, забезпечуючи цілісність і безпеку даних.

Шаблони проектування:

- MVC (Model-View-Controller): Шаблон MVC використовується для розділення логіки програми, інтерфейсу користувача та управління даними. Це дозволяє розробляти кожен компонент окремо, що спрощує процес тестування та підтримки програми. Модель (Model) відповідає за доступ до даних реєстру та їх обробку, представлення (View) відображає ці дані користувачеві, а контролер (Controller) обробляє вхідні запити від користувача та взаємодіє з моделлю та представленням для виконання відповідних дій.

- Singleton: Шаблон Singleton використовується для забезпечення єдиного доступу до бази даних резервних копій реєстру. Це гарантує, що в програмі існує лише один екземпляр об'єкта, який керує доступом до бази даних, що забезпечує узгодженість та запобігає конфліктам під час паралельного доступу.

Життєвий цикл ПЗ:

Модель життєвого циклу програмного забезпечення "ErgonReg" обґрунтована використанням гнучкої методології розробки (Agile). Agile підхід дозволяє швидко адаптувати програму до змінних вимог користувачів та

забезпечує постійне вдосконалення продукту через регулярні ітерації. Кожна ітерація включає планування, розробку, тестування та оцінку, що забезпечує високу якість та відповідність програмного забезпечення потребам користувачів. Agile також сприяє тісній співпраці з користувачами та врахуванню їх відгуків на всіх етапах розробки, що робить процес розробки більш гнучким та ефективним.

2.4. Опис використаних технологій та мов програмування

Технології:

- .NET Framework: Я використали класи з простору імен Microsoft.Win32 також інші для роботи з реєстром Windows, такі як RegistryKey для відкриття та маніпуляції ключами реєстру.

- WinForms: Я створили головну форму Form1, яка містить TreeView для відображення структури реєстру та DataGridView для показу значень вибраного ключа. Також були реалізовані контекстні меню (ContextMenuStrip) для швидкого доступу до функцій редагування реєстру.

- SQLite: Я реалізував методи LoadRegistryValues() для завантаження значень ключа реєстру, SaveRegistryToSQLite() для створення резервних копій, та LoadRegistryFromSQLite() для відновлення даних з резервної копії. Це легка реляційна база даних, яка не потребує окремого серверного програмного забезпечення, що робить її ідеальною для інтеграції у цю програму. Крім того, SQLite забезпечує високу продуктивність і невеликі вимоги до ресурсів, що є важливим для роботи на персональних комп'ютерах з різними технічними характеристиками.

Мови програмування:

-C#: Основна мова програмування для реалізації логіки програми. C# вибрано завдяки його тісній інтеграції з .NET Framework, високій продуктивності та зручності у використанні. На C# реалізовані всі основні функції програми, включаючи доступ до реєстру, обробку користувацьких запитів, створення резервних копій, виявлення та виправлення помилок, а також

оптимізацію реєстру. C# також забезпечує високу безпеку коду та підтримку сучасних об'єктно-орієнтованих принципів програмування, що робить розробку надійною та ефективною.

- XAML: Використовується для розробки інтерфейсу користувача. XAML (Extensible Application Markup Language) дозволяє описувати інтерфейс користувача у декларативний спосіб, що робить процес розробки більш структурованим та зрозумілим. Застосування XAML спрощує створення складних інтерфейсів, забезпечуючи чіткий поділ між логікою програми та її виглядом. У програмі "ErgonReg" XAML використовується для опису компонентів інтерфейсу, таких як форми, кнопки, поля введення, списки та інші елементи керування, що робить інтерфейс інтуїтивно зрозумілим та зручним для користувачів.

Деталізація по відношенню саме до "ErgonReg":

- Інтеграція з Windows API: Для доступу до реєстру використовується Windows API, що забезпечує безпосередню взаємодію з системними компонентами Windows. Це дозволяє програмі "ErgonReg" виконувати операції на рівні системи, такі як зчитування та запис ключів і значень реєстру, а також моніторинг змін у реєстрі.

- Модуль резервного копіювання: Окремий модуль програми відповідає за автоматичне резервне копіювання реєстру перед внесенням змін. Це забезпечує можливість швидкого відновлення попереднього стану реєстру у разі виникнення помилок або некоректних змін. Резервні копії зберігаються у базі даних SQLite, що гарантує їх доступність та цілісність.

- Алгоритми оптимізації: У програмі реалізовані спеціалізовані алгоритми для оптимізації реєстру. Вони аналізують структуру реєстру, виявляють зайві та застарілі записи, видаляють їх та виконують дефрагментацію, що покращує швидкість доступу до даних та загальну продуктивність системи.

- Інтерфейс користувача: Інтерфейс програми створено з урахуванням принципів ергономіки та зручності використання. Він включає інструменти для швидкого доступу до основних функцій, контекстне меню для роботи з ключами

та значеннями реєстру, а також візуальні підказки, що допомагають користувачам орієнтуватися у програмі.

Таким чином, використані технології та мови програмування забезпечують високу продуктивність, надійність та зручність у використанні програмного забезпечення "ErgonReg" для ергономічного налаштування реєстру операційної системи Windows.

2.5. Опис структури програми та алгоритмів її функціонування

Структура програми (Рис 1.8):

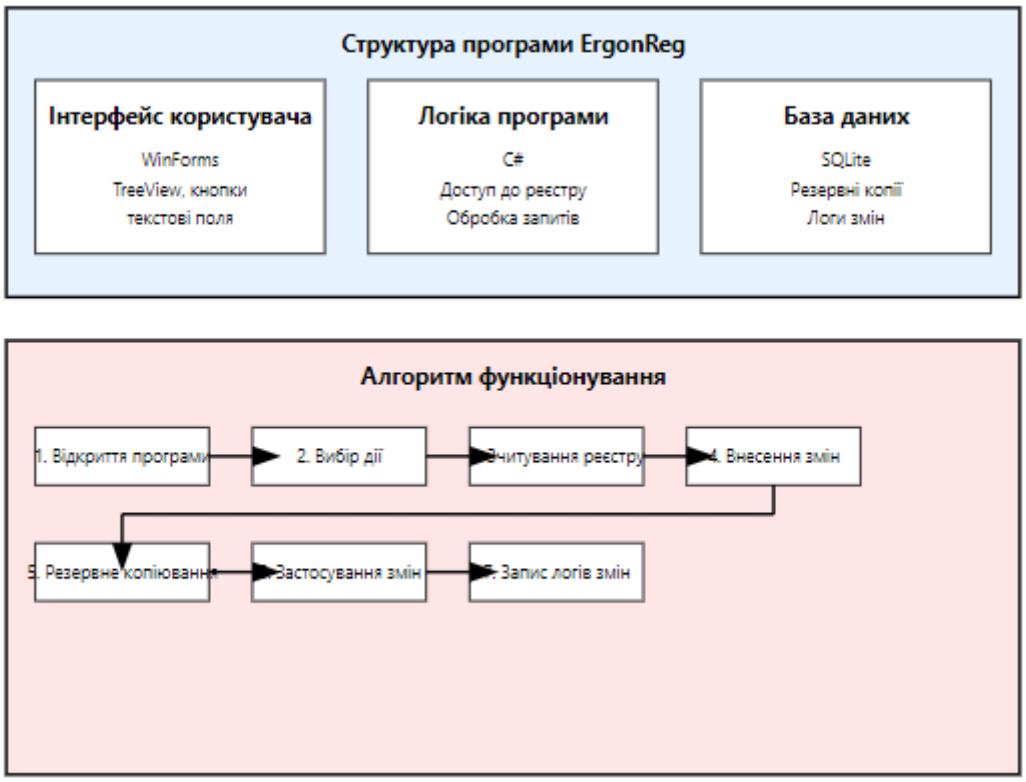

Рис. 1.8. Схема відображає основні алгоритми та структуру.

Програма "ErgonReg" складається з наступних основних компонентів:

- Інтерфейс користувача (UI): Розроблений з використанням WinForms, надає користувачам інтуїтивно зрозумілі інструменти для перегляду, редагування, видалення та створення ключів і значень реєстру. Інтерфейс включає елементи керування, такі як TreeView для відображення структури

реєстру, кнопки для виконання основних дій, текстові поля для введення нових значень та інформаційні вікна для підтвердження операцій.

- Логіка програми: Основна логіка програми написана на C#, що забезпечує високу продуктивність і стабільність роботи. Логіка включає функції для доступу до реєстру, обробки запитів користувачів, створення резервних копій, виявлення та виправлення помилок, а також оптимізації реєстру. Вона також відповідає за взаємодію між користувацьким інтерфейсом та базою даних.

- База даних: SQLite використовується для зберігання резервних копій реєстру та логів змін. Це дозволяє забезпечити надійне та швидке зберігання даних без необхідності використання окремого серверного програмного забезпечення. База даних зберігає всі зміни, внесені до реєстру, що дозволяє легко відновити попередній стан у разі помилки або потреби.

Алгоритм функціонування:

1. Відкриття програми: Користувач відкриває програму "ErgonReg". Програма ініціалізує всі необхідні компоненти, підключається до бази даних та завантажує поточний стан реєстру.

2. Вибір дії: Користувач вибирає необхідну дію з меню (перегляд, редагування, резервне копіювання). Інтерфейс програми оновлюється відповідно до вибраної дії.

3. Зчитування стану реєстру: Програма зчитує поточний стан реєстру та відображає його у вигляді дерева (TreeView). Користувач може навігувати по дереву, переглядаючи структуру та значення ключів.

4. Внесення змін: Користувач вносить зміни до реєстру через інтерфейс (наприклад, додає новий ключ, змінює значення існуючого ключа або видаляє ключ). Програма перевіряє введені дані на коректність.

5. Резервне копіювання: Перед застосуванням змін програма автоматично створює резервну копію поточного стану реєстру та зберігає її в базі даних SQLite. Це забезпечує можливість швидкого відновлення у разі виникнення проблем.

6. Застосування змін: Програма застосовує зміни до реєстру, оновлюючи відповідні ключі та значення. Інтерфейс користувача оновлюється для відображення нових даних.

7. Запис логів змін: Програма записує лог змін до бази даних, зберігаючи інформацію про внесені зміни, включаючи час, тип змін та значення до та після змін.

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Функціональні зв'язки між вхідними і вихідними даними програми:

1.Зчитування та відображення даних реєстру:

- Вхідні дані: Структура реєстру Windows, отримана через Windows API.
- Обробка: Програма зчитує структуру реєстру та перетворює її у деревоподібну структуру для відображення у TreeView.
- Вихідні дані: Візуальне представлення структури реєстру у інтерфейсі користувача.

2. Редагування значень реєстру:

- Вхідні дані: Користувацький ввід нових значень для ключів реєстру.
- Обробка: Програма валідує введені дані та застосовує їх до реєстру через Windows API.
- Вихідні дані: Оновлені значення в реєстрі Windows та відображення змін у інтерфейсі.

3. Створення резервних копій:

- Вхідні дані: Поточний стан реєстру перед внесенням змін.
- Обробка: Програма серіалізує дані реєстру та зберігає їх у базі даних SQLite.
- Вихідні дані: Резервна копія реєстру у базі даних SQLite.

4. Логування змін:

- Вхідні дані: Інформація про виконані користувачем дії з реєстром.
- Обробка: Програма форматує інформацію про зміни у структурований запис.
- Вихідні дані: Записи логів у текстовому форматі та в базі даних SQLite.

5. Відновлення з резервної копії:
   - Вхідні дані: Вибрана користувачем резервна копія з бази даних SQLite.
   - Обробка: Програма десеріалізує дані з бази даних та застосовує їх до реєстру Windows.
   - Вихідні дані: Відновлений стан реєстру Windows та оновлене відображення у інтерфейсі.
6. Оптимізація реєстру:
   - Вхідні дані: Поточний стан реєстру Windows.
   - Обробка: Програма аналізує структуру реєстру, виявляє надлишкові або застарілі записи.
   - Вихідні дані: Оптимізована структура реєстру Windows та лог виконаних змін.

2.7. Опис розробленого програмного продукту

2.7.1. Використані технічні засоби:

- **ПК**: Персональні комп'ютери з операційною системою Windows.
- **Процесор**: Мінімум 1.5 GHz.
- **Оперативна пам'ять**: Мінімум 2 GB.
- **Дисковий простір**: Мінімум 100 MB для встановлення програми.
- **Операційна система**: Windows 7 або новіша.

2.7.2. Використані програмні засоби:

- **Операційна система:** Windows.
- **.NET Framework**: Для функціонування інтерфейсу користувача.

- **SQLite**: Для зберігання резервних копій та логів змін.

- **C#**: Основна мова програмування.

2.7.3. Виклик та завантаження програми:

Програмне забезпечення встановлюється на комп'ютер користувача за допомогою інсталяційного пакету, який включає всі необхідні файли та компоненти для коректної роботи програми. Після завершення процесу встановлення програма готова до використання (Рис 1.9).

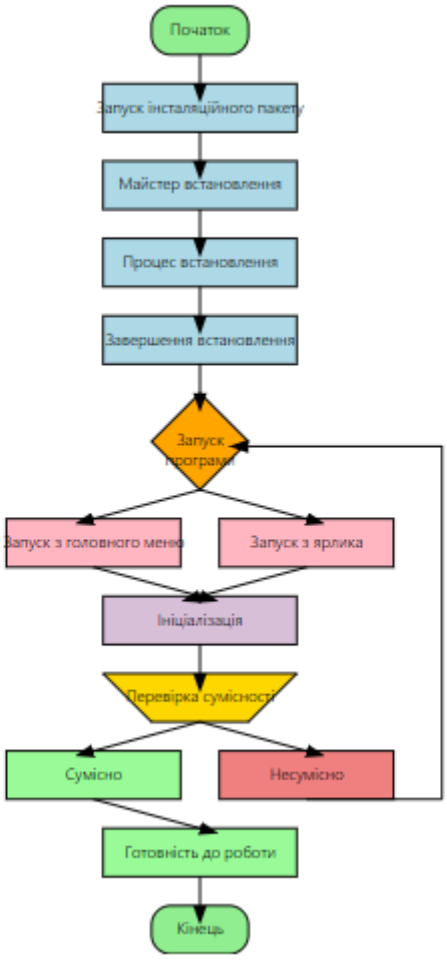

Рис 1.9. Блок-схема ілюструє процес виклику та завантаження програми

Процес встановлення:

1. Запуск інсталяційного пакету: Користувач запускає інсталяційний файл (наприклад, `ErgonRegSetup.exe`), після чого відкривається майстер встановлення.

2. Майстер встановлення: Користувач проходить через кроки майстра, погоджуючись з ліцензійною угодою, вибираючи місце встановлення та інші параметри.

3. Процес встановлення: Програма встановлюється на комп'ютер користувача, копіюючи необхідні файли та налаштування.

4. Завершення встановлення: Після успішного завершення встановлення, користувач бачить відповідне повідомлення та може вибрати опцію для автоматичного запуску програми після закриття майстра встановлення.

Запуск програми:

1. Запуск з головного меню: Після встановлення, програма "ErgonReg" додається до головного меню Windows. Користувач може знайти її в списку програм та запустити.

2. Запуск з ярлика на робочому столі: Інсталяційний пакет створює ярлик на робочому столі для швидкого доступу до програми. Користувач може запустити програму, двічі клацнувши на ярлику.

Первинний запуск програми:

1. Ініціалізація: При першому запуску програма виконує первинну ініціалізацію, що включає перевірку сумісності з операційною системою Windows, налаштування початкових параметрів та створення необхідних каталогів і файлів.

2. Перевірка сумісності: Програма перевіряє, чи відповідає поточна версія операційної системи Windows мінімальним вимогам для роботи "ErgonReg". Якщо виявлено несумісність, користувач отримує відповідне повідомлення з інструкціями щодо вирішення проблеми.

3. Готовність до роботи: Після завершення ініціалізації та перевірок програма готова до використання. Відкривається головне вікно програми з інтуїтивно зрозумілим інтерфейсом, який дозволяє користувачам одразу почати роботу з реєстром.

Ці графічні матеріали та пояснення ілюструють процес встановлення та первинного запуску програмного забезпечення "ErgonReg", показуючи, як

користувач може легко встановити та запустити програму для роботи з реєстром операційної системи Windows.

2.7.4. Опис інтерфейсу користувача:

Інтерфейс користувача програмного забезпечення розроблений з урахуванням принципів ергономіки та зручності використання. Він інтуїтивно зрозумілий і дозволяє користувачам швидко виконувати необхідні операції з реєстром Windows.

Основні елементи інтерфейсу:

- Головне меню: Розташоване у верхній частині вікна програми, головне меню надає доступ до основних функцій програми, таких як перегляд реєстру, редагування, резервне копіювання та відновлення. Меню розділене на декілька вкладок, що спрощує навігацію та забезпечує швидкий доступ до всіх можливостей програми.

- Панель інструментів: Розташована під головним меню, панель інструментів містить кнопки для швидкого доступу до основних операцій, таких як створення нового ключа, редагування, видалення та збереження змін. Кожна кнопка має відповідну піктограму, що робить інтерфейс більш зрозумілим.

- Вікно перегляду реєстру: Основне вікно програми, де відображається структура реєстру у вигляді дерева (TreeView). Користувачі можуть розгортати та згортати гілки дерева для перегляду підключів та значень. Вікно перегляду також підтримує функції фільтрації та пошуку, що дозволяє швидко знаходити необхідні ключі та значення (Рис 2.1).

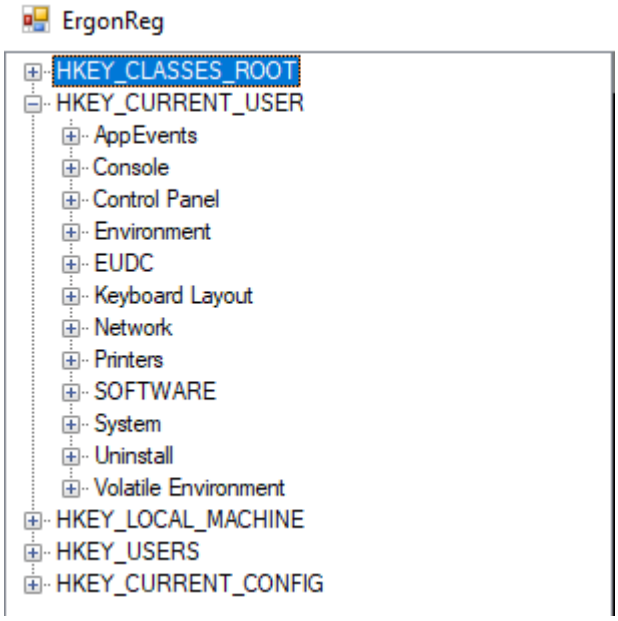

Рис. 2.1. Структура реєстру у вигляді дерева

- Діалогові вікна: Використовуються для підтвердження дій користувача та відображення повідомлень про помилки. Наприклад, перед видаленням ключа або значення з'являється діалогове вікно з запитом на підтвердження дії. У випадку виникнення помилки, користувач отримує детальне повідомлення з описом проблеми та можливими способами її вирішення (Рис 2.2).

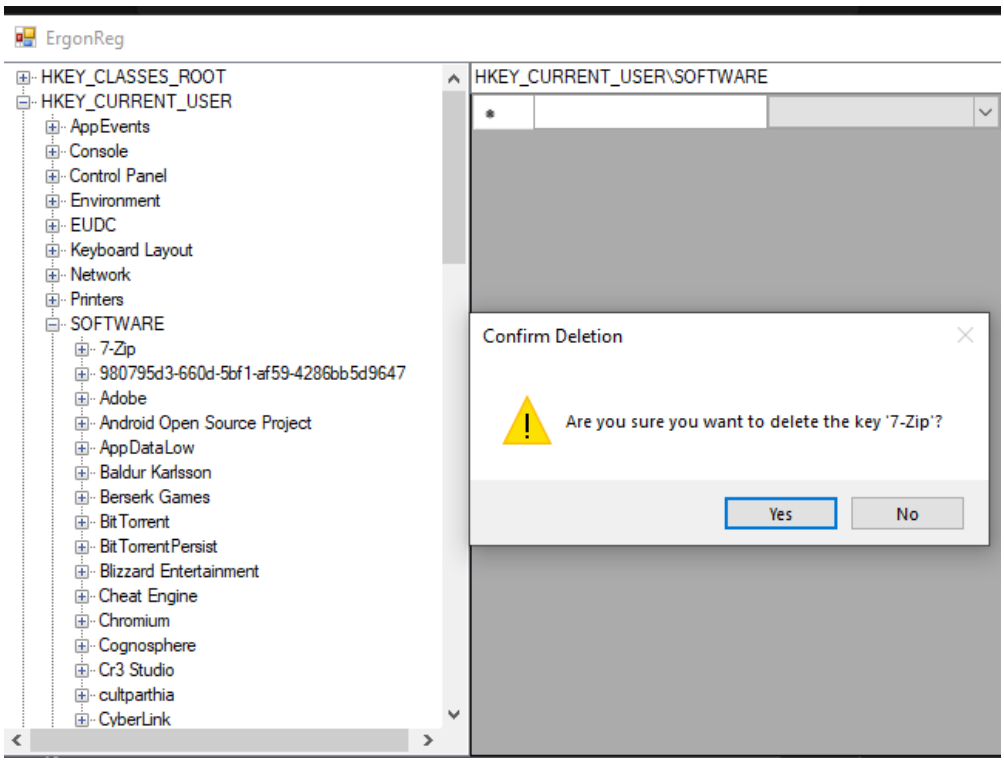

Рис. 2.2. Діалогове вікно

- Інтерактивна довідкова система: Програма включає інтерактивну довідкову систему, яка допомагає користувачам швидко освоїти основні функції та операції. Довідкова система доступна з головного меню та містить детальні інструкції з використання всіх можливостей програми, а також відповіді на найпоширеніші запитання.

Приклади використання інтерфейсу:

1. Перегляд структури реєстру: Користувач відкриває програму та бачить структуру реєстру у вигляді дерева. Він може розгортати гілки дерева, щоб побачити підключі та значення. При виборі конкретного ключа, у правій частині вікна відображаються деталі цього ключа (Рис 2.3).

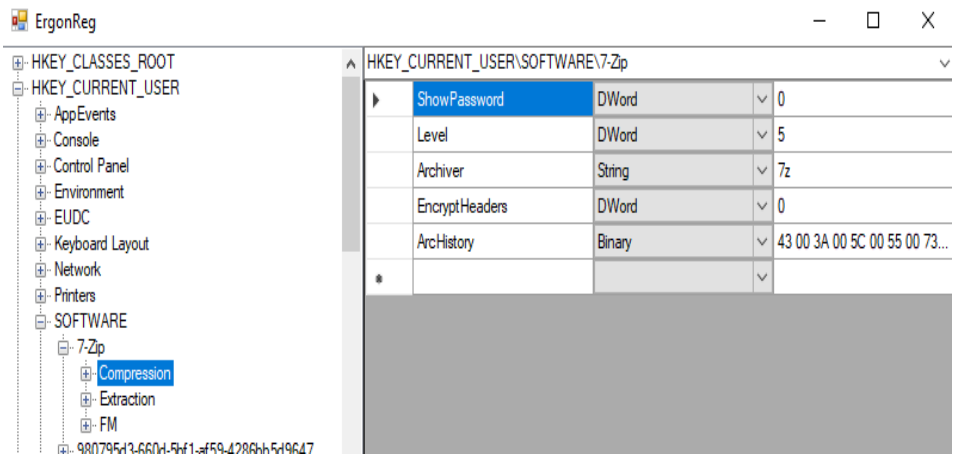

Рис. 2.3. Значення вибраного ключа

2. Редагування значення реєстру: Користувач вибирає ключ у дереві та редагує на заняченння в правій частині. З'являється діалогове вікно з запитом на підтвердження дії. Після натискання кнопки "ОК" зміни застосовуються, і реєстр оновлюється (Рис 2.4).

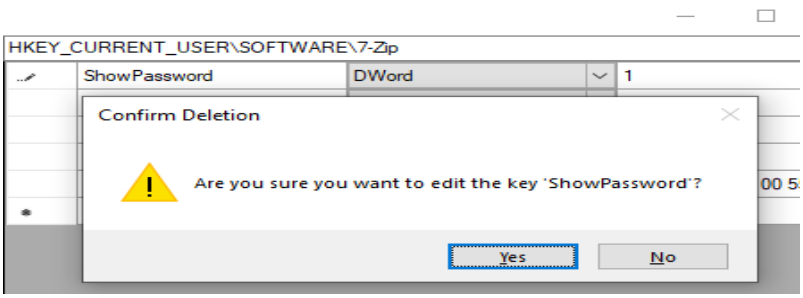

Рис. 2.4. Діалогове вікно

3. Створення резервної копії: Користувач натискає кнопку "Резервне копіювання" на панелі інструментів. Програма створює резервну копію поточного стану реєстру та зберігає її у базі даних SQLite. Користувач отримує повідомлення про успішне створення резервної копії (Рис 2.5, Рис 2.6).

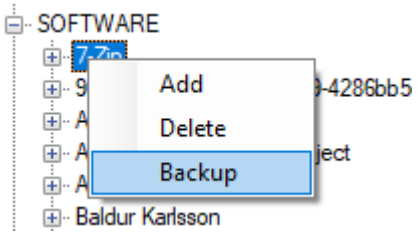

Рис. 2.5. Кнопка на панелі інструментів

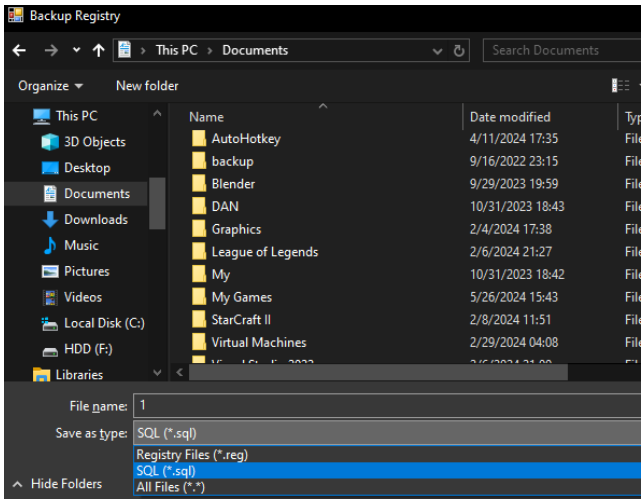

Рис. 2.6. Файлове діалогове вікно

4. Відновлення з резервної копії: Користувач натискає кнопку "Відновлення" на панелі інструментів та вибирає необхідну резервну копію з списку. Програма відновлює реєстр до попереднього стану, використовуючи вибрану копію, і користувач отримує повідомлення про успішне відновлення.

Програма "ErgonReg" забезпечує зручний, інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам легко керувати реєстром операційної системи Windows, виконуючи основні операції з мінімальними зусиллями та високою ефективністю.

Висновок

У другому розділі розглянуто процес проектування та розробки програмного забезпечення "ErgonReg" для налаштування та відкату реєстру персональних комп'ютерів.

Спочатку визначено функціональне та експлуатаційне призначення програми. "ErgonReg" надає зручний інтерфейс для перегляду, редагування, видалення та резервного копіювання реєстру, що допомагає оптимізувати роботу операційної системи та підвищити її стабільність.

Описано, що складні математичні методи не використовуються, лише стандартні алгоритми обробки та аналізу даних реєстру.

Розглянуто клієнт-серверну архітектуру програми, використання шаблонів MVC та Singleton, а також методологію розробки Agile.

Використані технології: .NET Framework, WinForms та SQLite. Основна мова програмування - C#, інтерфейс створено з використанням XAML.

Детально описано структуру програми, алгоритми функціонування, обробку вхідних та вихідних даних.

Нарешті, надано інформацію про технічні та програмні засоби, спосіб виклику та завантаження програми, а також інтерфейс користувача.

Таким чином, розділ охоплює всі аспекти проектування та розробки "ErgonReg", забезпечуючи його якість, надійність та зручність для користувачів.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. q – передбачуване число операторів програми – 940;
2. C – коефіцієнт складності програми – 0,95;
3. p – коефіцієнт корекції програми в ході її розробки – 0,3;
4. $C_{пр}$ – годинна заробітна плата програміста – 200 грн/год;
5. B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,3;
6. k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,25;
7. $C_{мв}$ – вартість машино-години ЕОМ – 4,84 грн/год;
8. B_k – число виконавців – 1 людина;
9. F_p – місячний фонд робочого часу (при 40 годинному робочому тижні) – 176 годин.

Примітка:

У 2023 році заробітна плата програміста кваліфікації Junior Software Engineer, який працює з мовою програмування C#, становить 600$ на місяць. Враховуючи те, що програміст працює в середньому 22 дні на місяць при чинному курсі Національного банку України 1 долар = 38 гривень, заробітна плата програміста за годину виходить 140 грн/год.

Вихідна вартість машино-години включає в себе витрати на електроенергію, доступ до мережі інтернет, вартість амортизації приладу.

Зарядний блок живлення сучасного ЕОМ, в даному випадку – комп’ютер та монітор, розрахований на потужність до 135 Вт.

Тариф на електроенергію: 4,32 грн/кВт·год.

Ціна за годину:

$$4{,}32 * 0{,}135 \text{ кВт} = 0{,}58 \text{ грн/год}$$

Тариф інтернет зв'язку: 190 грн/місяць

Ціна за годину:

$$190 / (176 \text{ год.}) \approx 1{,}08 \text{ грн/год.}$$

Погодинна амортизація компютера за умови початкової вартості 40 000 грн., ліквідаційної вартості – 10 000 грн, терміну корисного використання – 4 роки прямолінійним методом розраховується відповідно:

$$(40000 \text{ грн} – 5000 \text{ грн}) / (12 \text{ міс} \cdot 4 \text{ роки} \cdot 176 \text{ годин}) \approx 4{,}14 \text{ грн}$$

Сумарна вартість машино-години ЕОМ складає:

$$0{,}58 + 1{,}08 + 4{,}14 = 5{,}8 \text{ грн/год.}$$

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ люд-годин,} \qquad (3.1)$$

- де t_o-витрати праці на підготовку й опис поставленої задачі (приймається 45 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a- витрати праці на розробку блок-схеми алгоритму;

t_n-витрати праці на програмування по готовій блок-схемі;

$t_{отл}$-витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q=q\cdot C(1+p) \qquad (3.2)$$

де:

- q - передбачуване число операторів (940);

- C - коефіцієнт складності програми (0,95);

- p - коефіцієнт корекції програми в ході її розробки (0,3).

Звідси умовне число операторів в програмі:

$$Q = 940\cdot 0{,}95\cdot(1 + 0{,}3) = 1160$$

Витрати праці на вивчення опису задачі t_u визначаються з урахуванням уточнення опису і кваліфікації програміста:

$$t_u=\frac{Q\cdot B}{(75...85)K}, \text{ людино-годин} \qquad (3.3)$$

де:

- B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1,3);

- k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності(1,25).

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1,3). З урахуванням коефіцієнта кваліфікації k = 1,25 отримуємо витрати праці на вивчення опису завдання:

$$t_u= (1160\cdot 1{,}3) / (75\cdot 1{,}25) = 16 \text{ людино-годин}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20...25)K} \quad (3.4)$$

де Q – умовне число операторів програми;

K – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу, отримаємо:

$$t_a = 1160/ (20 \cdot 1,25) = 46 \text{ людино-годин}$$

Витрати праці на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \quad (3.5)$$

Підставивши відповідні значення в формулу, отримаємо:

$$t_n = 1160/ (20 \cdot 1,25) = 46 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- За умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)K} \quad (3.6)$$

$$t_{отл} = 1160/ (4 \cdot 1,25) = 232 \text{ людино-годин}$$

- За умови комплексного налагодження завдання:

$$t^к_{отл} = 1,5 \cdot t_{отл} \quad (3.7)$$

$$t^{к}_{отл} = 1,5 \cdot 232 = 348 \text{ людино-годин}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_д = t_{др} + t_{до} \quad (3.8)$$

- де $t_{др}$-трудомісткість підготовки матеріалів і рукопису.

- $t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{др} = \frac{Q}{(15\ldots20)K} \quad (3.9)$$

$t_{др} = 1160/ (15 \cdot 1,25) = 62$ людино-годин.

- $t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0,75 \cdot t_{др} \quad (3.10)$$

$t_{до} = 0,75 \cdot 62 = 46$ людино-годин.

Розрахуємо витрати праці на підготовку документації за формулою (3.8):

$$t_д = 62 + 46 = 108 \text{ людино-годин.}$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t = 45 + 16 + 46 + 46 + 232 + 108 = 493 \text{ людино-години}$$

У результаті отримано, що в загальній складності необхідно 493 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО}=З_{ЗП}+З_{МВ}, \text{ грн}, \quad (3.11)$$

де $З_{ЗП}$– заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП}=t\cdot С_{ПР}, \text{ грн}, \quad (3.12)$$

де: t - загальна трудомісткість, людино-годин

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 200 грн / год, отримуємо:

$$З_{ЗП}= 493 \cdot 200 = 98600 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{МВ} = t_{отл} \cdot СМ, \text{ грн}, \quad (3.13)$$

де: $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год

- $С_{мч}$ - вартість машино-години ЕОМ, грн/год (5,8 грн/год).

Підставивши в формулу (3.13) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{МВ}= 232 \cdot 5{,}8 = 1345{,}6 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО}= 98600 + 1345{,}6 = 99945{,}6 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T=\frac{t}{B_k \cdot F_p}, \text{ мес.} \qquad (3.14)$$

де:

- B_k- число виконавців (дорівнює 1);

- F_p - місячний фонд робочого часу (при 40-годинному робочому тижні (Fp = 176 годин).

Звідси очікуваний період створення ПЗ:

$$T= 493 / 1 \cdot 176 \approx 2{,}8 \text{ міс.}$$

Висновки: трудомісткість розробленого веб-застосунку для інтернет-магазину з продажу гелієвих кульок становить 493 людино-години. Обчислена вартість роботи по створенню програми дорівнює 99945,6 гривень. Визначений затрачений час на створення програмного забезпечення становить 2,8 місяці.

ВИСНОВОК

У результаті виконання кваліфікаційної роботи було успішно досягнуто поставленої мети — розроблено програмне забезпечення для ергономічного налаштування реєстру операційної системи Windows на персональних комп'ютерах. Створений програмний продукт забезпечує можливість безпечного та ефективного редагування системного реєстру, оптимізації його роботи та усунення потенційних проблем, пов'язаних з некоректним налаштуванням реєстру.

Розроблена програма являє собою комплексну систему для управління налаштуваннями реєстру Windows, яка включає наступні реалізовані функції:

- Перегляд та аналіз вмісту реєстру.
- Безпечне редагування ключів і значень реєстру.
- Автоматичне резервне копіювання реєстру перед внесенням змін.
- Виявлення та виправлення помилок у реєстрі.
- Оптимізація реєстру для підвищення продуктивності системи.

Актуальність роботи визначається зростаючою потребою в забезпеченні стабільної та продуктивної роботи операційної системи Windows шляхом належного налаштування реєстру. У сучасному світі, де комп'ютери є невід'ємною частиною професійної та особистої діяльності, розроблений програмний продукт є важливим інструментом для забезпечення ефективного управління системними налаштуваннями. Надання зручних інструментів для редагування та оптимізації реєстру дозволяє користувачам уникати потенційних проблем та підвищувати загальну продуктивність комп'ютера.

Створена програма призначена для використання широким колом користувачів, включаючи IT-фахівців, системних адміністраторів та звичайних користувачів, які прагнуть покращити роботу своїх персональних комп'ютерів.

Даний програмний продукт спрощує процес налаштування реєстру, дозволяє користувачам швидко і легко виконувати необхідні дії з налаштування

та оптимізації системи, а також забезпечує безпеку та цілісність даних. Структура програми включає веб-застосунок, написаний мовою програмування C++ з використанням фреймворку .NET для створення інтерфейсу користувача та системи керування базами даних SQLite для зберігання резервних копій та логів змін.

В «Економічному розділі» було визначено трудомісткість програми (296,83 людино-години), обчислені витрати на створення застосунку (81907,5 грн.) та затрачений час на розробку 3,63 місяці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Russinovich M., Solomon D. A., Ionescu A. Windows Internals, Part 1: System architecture, processes, threads, memory management, and more. 7th ed. Redmond: Microsoft Press, 2021. 800 p.

2. Tanenbaum A. S., Bos H. Modern operating systems. 4th ed. Boston : Pearson, 2015. 1136 p.

3. Фролов А. В., Фролов Г. В. Операційні системи: архітектура, алгоритми, проектування застосунків. Дніпро : ДНУ, 2019. 320 с.

4. Robins G. The Windows Registry: A Forensics Resource. SANS Institute, 2000. URL: https://www.sans.org/reading-room/whitepapers/forensics/windows-registry-forensics-resource-197 (дата звернення: 01.07.2024).

5. Цилюрик Г. І., Цилюрик С. М. Операційні системи: архітектура та функціонування. Вінниця : ВНТУ, 2018. 212 с.

6. Solomon D. A., Russinovich M. E. Windows Internals, Part 1. 6th ed. Redmond : Microsoft Press, 2012. 800 p.

7. Tanenbaum A. S., Woodhull A. S. Operating Systems: Design and Implementation. 3rd ed. Upper Saddle River : Prentice Hall, 2006. 1054 p.

8. Silberschatz A., Galvin P. B., Gagne G. Operating System Concepts. 10th ed. Hoboken : Wiley, 2018. 1040 p.

9. Жежнич П. І., Жежнич Н. В. Операційні системи. Вінниця : ВНТУ, 2019. 256 с.

10. Stallings W. Operating Systems: Internals and Design Principles. 9th ed. Boston : Pearson, 2015. 800 p.

11. Bovet D. P., Cesati M. Understanding the Linux Kernel. 3rd ed. Sebastopol : O'Reilly Media, 2005. 944 p.

12. Гаркуша І. М. Конспект лекцій з дисципліни "Операційні системи" для студентів галузі знань 12 "Інформаційні технології". Дніпро, 2020. 150 с.

13. Яремчук Ю. Є., Яремчук О. Ю. Операційні системи: базові структури та принципи функціонування. Вінниця : ВНТУ, 2020. 280 с.

14. Solomon D. A., Russinovich M. E. Windows Internals. 5th ed. Redmond : Microsoft Press, 2009. 1232 p.

15. Love R. Linux Kernel Development. 3rd ed. Upper Saddle River : Addison-Wesley Professional, 2010. 440 p.

16. Stallings W. Operating Systems: Internals and Design Principles. 9th ed. London: Pearson, 2020. 800 p.

17. Yosifovich P., Ionescu A., Russinovich M. E., Solomon D. A. Windows Internals, Part 2. 7th ed. Redmond: Microsoft Press, 2021. 896 p.

18. Чумаченко В. В., Головатий А. В. Операційні системи: принципи побудови та функціонування. Київ : КПІ ім. Ігоря Сікорського, 2019. 240 с.

19. Microsoft. Windows Registry Documentation. Microsoft Learn, 2023. URL: https://learn.microsoft.com/en-us/windows/win32/sysinfo/registry (дата звернення: 01.07.2024)

20. Згуровський М. З., Ковтун І. М. Операційні системи: принципи організації та функціонування. Київ : КПІ ім. Ігоря Сікорського, 2020. 300 с.

21. Database. Wikipedia. URL: https://en.wikipedia.org/wiki/Database (дата звернення: 01.07.2024).

22. Perlis A. J. The synthesis of algorithmic systems. Journal of the ACM. 1966. Vol. 13, No. 1. P. 1-9.

23. File system. Wikipedia. URL: https://en.wikipedia.org/wiki/File_system (дата звернення: 01.07.2024).

24. Application software. Wikipedia. URL: https://en.wikipedia.org/wiki/Application_software (дата звернення: 01.07.2024).

25. Windows 10. Wikipedia. URL: https://en.wikipedia.org/wiki/Windows_10 (дата звернення: 01.07.2024).

Додаток А

ЛІСТИНГ ПРОГРАМИ

Form1.cs:

```
        using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Diagnostics;
using System.Linq;
using System.Security;
using System.Security.Permissions;
using System.Windows.Forms;
using System.Xml.Linq;
using Microsoft.Data.Sqlite;
using System.IO;

namespace RegistryEditor
{
   public partial class Form1 : Form
   {
      private RegistryKey currentKey;

      public Form1()
      {
         InitializeComponent();
         LoadRegistryKeys();

         // Заповнюємо комбобокс типами реєстру
         gridType.Items.AddRange(Enum.GetNames(typeof(RegistryValueKind)));
      }
```

```
// Завантаження кореневих ключів реєстру
private void LoadRegistryKeys()
{
    // Очищає вузли TreeView
    treeView1.Nodes.Clear();
    // Додає до нього кореневі ключі з реєстру Windows, наприклад HKEY_CLASSES_ROOT, HKEY_CURRENT_USER, HKEY_LOCAL_MACHINE, HKEY_USERS і HKEY_CURRENT_CONFIG.
    AddRootKey("HKEY_CLASSES_ROOT", Registry.ClassesRoot);
    AddRootKey("HKEY_CURRENT_USER", Registry.CurrentUser);
    AddRootKey("HKEY_LOCAL_MACHINE", Registry.LocalMachine);
    AddRootKey("HKEY_USERS", Registry.Users);
    AddRootKey("HKEY_CURRENT_CONFIG", Registry.CurrentConfig);
}

// Додавання кореневого ключа до TreeView
private void AddRootKey(string name, RegistryKey key)
{
    // Він створює TreeNode із вказаним іменем, призначає об’єкт RegistryKey його тегу.
    TreeNode rootNode = new TreeNode(name);
    rootNode.Tag = key;
    // Додає його до колекції «Вузли» TreeView, а потім до кореневого вузла додає дочірній вузол із текстом «Завантаження...».
    treeView1.Nodes.Add(rootNode);
    rootNode.Nodes.Add(new TreeNode("Loading..."));
}

// Обробка події розгортання вузла дерева
private void treeView1_BeforeExpand(object sender, TreeViewCancelEventArgs e)
```

```
    {
        // Він перевіряє, чи має вузол дочірні елементи з текстом «Завантаження...», і якщо так, він викликає метод getRegistrySubKeys для завантаження підключів.
        if (e.Node.Nodes.Count == 1 && e.Node.Nodes[0].Text == "Loading...")
        {
            getRegistrySubKeys(e.Node);
        }
        // Потім він оновлює comboBox повним шляхом до реєстру вибраного вузла
        comboBox1.Items.Clear();
        comboBox1.Text = GetFullRegistryPath(e.Node);
        // Заповнює comboBox шляхами дочірніх вузлів.
        foreach (TreeNode node in e.Node.Nodes)
        {
            comboBox1.Items.Add(comboBox1.Text + "\\" + node.Text);
        }
    }

    // Отримання підключів реєстру для вузла
    private void getRegistrySubKeys(TreeNode parentNode)
    {
        // Очищає дочірні вузли даного parentNode у перегляді дерева.
        parentNode.Nodes.Clear();
        // Він отримує підрозділи розділу реєстру, пов’язаного з parentNode.
        RegistryKey parentKey = parentNode.Tag as RegistryKey;

        if (parentKey == null)
            return;

        try
```

```
        {
            // Потім заповнює перегляд дерева підключами як дочірніми вузлами.
            foreach (string subKeyName in parentKey.GetSubKeyNames())
            {
                try
                {
                    RegistryKey subKey = parentKey.OpenSubKey(subKeyName, true);
                    if (subKey != null)
                    {
                        TreeNode subNode = new TreeNode(subKeyName);
                        subNode.Tag = subKey;
                        parentNode.Nodes.Add(subNode);
                        subNode.Nodes.Add(new TreeNode("Loading..."));
                    }
                }
                // Він також обробляє винятки, пов’язані з доступом до розділів реєстру.
                catch (System.Security.SecurityException)
                {
                    TreeNode errorNode = new TreeNode(subKeyName + " (Access Denied)");
                    parentNode.Nodes.Add(errorNode);
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error loading subkeys: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
```

```
// Обробка вибору вузла в TreeView
private void treeView1_AfterSelect(object sender, TreeViewEventArgs e)
{
    // Завантажуємо значення реєстру для вибраного вузла
    LoadRegistryValues(e.Node);
}

// Завантаження значень реєстру для вибраного вузла
private void LoadRegistryValues(TreeNode node)
{
    // Очищаємо таблицю
    dataGridView1.Rows.Clear();
    // Зберігаємо вибраний вузол
    currentKey = node.Tag as RegistryKey;
    if (currentKey == null)
        return;

    try
    {
        // Завантажуємо значення
        foreach (string valueName in currentKey.GetValueNames())
        {
            // Зберігаємо значення
            object value = currentKey.GetValue(valueName);
            // Зберігаємо тип
            RegistryValueKind valueKind = currentKey.GetValueKind(valueName);
            // Конвертуємо значення
            string valueString = ConvertRegistryValue(value, valueKind);
            // Додаємо значення в таблицю
```

```
                dataGridView1.Rows.Add(valueName, valueKind.ToString(),
valueString);
            }
        }
        catch (Exception ex)
        {
            // Виводимо помилку
            MessageBox.Show("Error loading registry values: " + ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    // Конвертація значення реєстру в рядок для відображення
    private string ConvertRegistryValue(object value, RegistryValueKind
valueKind)
    {
        switch (valueKind)
        {
            case RegistryValueKind.Binary:
                return BitConverter.ToString((byte[])value).Replace("-", " ");
            case RegistryValueKind.MultiString:
                return string.Join(", ", (string[])value);
            case RegistryValueKind.ExpandString:
            case RegistryValueKind.String:
            case RegistryValueKind.DWord:
            case RegistryValueKind.QWord:
            case RegistryValueKind.Unknown:
            default:
                return value.ToString();
        }
    }
```

```
    // Конвертація рядка в значення реєстру відповідного типу
    private object ConvertToRegistryValue(string value, RegistryValueKind kind)
    {
        switch (kind)
        {
            case RegistryValueKind.DWord:
                return Convert.ToInt32(value);
            case RegistryValueKind.QWord:
                return Convert.ToInt64(value);
            case RegistryValueKind.Binary:
                return value.Split(' ').Select(s => Convert.ToByte(s, 16)).ToArray();
            case RegistryValueKind.MultiString:
                return value.Split(',').Select(s => s.Trim()).ToArray();
            default:
                return value;
        }
    }

    // Обробка кліку правою кнопкою миші на вузлі
    private void treeView1_NodeMouseClick(object sender,
TreeNodeMouseClickEventArgs e)
    {
        // Якщо натиснули правою кнопкою, встановлюємо вибраний вузол
        if (e.Button == MouseButtons.Right)
        {
            // Встановлюємо вибраний вузол
            treeView1.SelectedNode = e.Node;
        }
    }
```

```
// Отримання повного шляху реєстру для вузла
private string GetFullRegistryPath(TreeNode node)
{
    List<string> pathParts = new List<string>();
    // Поки вузол не є кореневим, додаємо його назву до шляху
    while (node != null)
    {
        // Додаємо назву вузла до шляху
        pathParts.Insert(0, node.Text);
        // Поки вузол не є кореневим
        node = node.Parent;
    }
    // Повертаємо шлях
    return string.Join("\\", pathParts);
}

// Видалення ключа реєстру
private void toolStripMenuItem1_Click(object sender, EventArgs e)
{
    TreeNode selectedNode = treeView1.SelectedNode;
    if (selectedNode == null || selectedNode.Parent == null)
    {
        // Помилка видалення кореневого вузла
        MessageBox.Show("Cannot delete root keys.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    // Підтвердження видалення
```

```
        if (MessageBox.Show($"Are you sure you want to delete the key '{selectedNode.Text}'?", "Confirm Deletion", MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Yes)
        {
            try
            {
                RegistryKey parentKey = selectedNode.Parent.Tag as RegistryKey;
                parentKey.DeleteSubKeyTree(selectedNode.Text);
                // Видаляємо вибраний вузол
                selectedNode.Remove();
                // Показуємо повідомлення про успішне видалення
                MessageBox.Show("Key deleted successfully.", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            catch (Exception ex)
            {
                // Показуємо помилку
                MessageBox.Show($"Error deleting key: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }

    // Створення резервної копії ключа реєстру
    private void toolStripMenuItem2_Click(object sender, EventArgs e)
    {
        TreeNode selectedNode = treeView1.SelectedNode;
        if (selectedNode == null)
        {
            // Помилка створення резервної копії
```

```
            MessageBox.Show("No key selected for backup.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }

        SaveFileDialog saveFileDialog = new SaveFileDialog
        {
            Filter = "Registry Files (*.reg)|*.reg|SQL (*.sql)|*.sql|All Files (*.*)|*.*",
            Title = "Backup Registry"
        };

        // Показуємо діалог збереження
        if (saveFileDialog.ShowDialog() == DialogResult.OK)
        {
            try
            {
                // Тип фільтруємо відповідно до вибраного типу файлу
                switch (saveFileDialog.FilterIndex)
                {
                    case 2:
                        // Створюємо резервну копію ключа в базу даних
                        SaveRegistryToSQLite(saveFileDialog.FileName);
                        break;
                    default:
                        // Створюємо резервну копію ключа в реєстр
                        SaveRegistryToReg(selectedNode, saveFileDialog.FileName);
                        break;
                }
                // Показуємо повідомлення про успішне збереження
                MessageBox.Show("Key backed up successfully.", "Success",
MessageBoxButtons.OK, MessageBoxIcon.Information);
```

```
            }
            catch (Exception ex)
            {
                // Показуємо помилку
                MessageBox.Show($"Error backing up key: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }

    private void SaveRegistryToReg(TreeNode selectedNode, string fileName)
    {
        // Створюємо резервну копію ключа в реєстр
        string fullPath = GetFullRegistryPath(selectedNode);
        // Створюємо аргументи команди
        string arguments = $"export \"{fullPath}\" \"{fileName}\"";
        // Виконуємо команду
        System.Diagnostics.Process.Start("reg.exe", arguments);
    }

    // Перезапуск програми з правами адміністратора
    private void RestartAsAdmin()
    {
        ProcessStartInfo processInfo = new ProcessStartInfo();
        processInfo.Verb = "runas"; // Це запитує права адміністратора
        // Встановлюємо шлях до цієї програми
        processInfo.FileName = Application.ExecutablePath;
        try
        {
            // Виконуємо запит
            Process.Start(processInfo);
```

```
            // Закриваємо програму
            Application.Exit();
        }
        catch (Win32Exception)
        {
            // Якщо ви не маєте права адміністратора
            MessageBox.Show("Operation cancelled. Some changes may not be possible without elevated permissions.", "Operation Cancelled", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }

    // Обробка зміни значення в DataGridView
    private void dataGridView1_CellValueChanged(object sender, DataGridViewCellEventArgs e)
    {
        if (e.RowIndex < 0 || currentKey == null) return;

        DataGridViewRow row = dataGridView1.Rows[e.RowIndex];
        string name = row.Cells["gridName"].Value?.ToString();
        string typeStr = row.Cells["gridType"].Value?.ToString();
        string valueS = row.Cells["gridValue"].Value?.ToString();

        if (string.IsNullOrEmpty(name) || string.IsNullOrEmpty(typeStr) || string.IsNullOrEmpty(valueS)) return;

        if (MessageBox.Show($"Are you sure you want to edit the key '{name}'?", "Confirm Deletion", MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Yes)
        {
            try
```

```
            {
                // Перетворюємо тип
                RegistryValueKind kind = (RegistryValueKind)Enum.Parse(typeof(RegistryValueKind), typeStr);
                // Перетворюємо значення
                object convertedValue = ConvertToRegistryValue(valueS, kind);
                // Змінюємо значення
                currentKey.SetValue(name, convertedValue, kind);
                // Показуємо повідомлення про успішне збереження
                MessageBox.Show("Value updated successfully.", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            catch (Exception ex)
            {
                // Показуємо помилку
                MessageBox.Show($"Error updating registry value: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        } else
        {
            // Видалення рядків
            dataGridView1.Rows.Clear();
            if (currentKey == null)
                return;

            try
            {
                foreach (string valueName in currentKey.GetValueNames())
                {
                    object value = currentKey.GetValue(valueName);
```

```
                RegistryValueKind valueKind = currentKey.GetValueKind(valueName);

                string valueString = ConvertRegistryValue(value, valueKind);

                dataGridView1.Rows.Add(valueName, valueKind.ToString(), valueString);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error loading registry values: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

// Обробка додавання нового рядка в DataGridView
private void dataGridView1_UserAddedRow(object sender, DataGridViewRowEventArgs e)
{
    int newRowIndex = e.Row.Index - 1;
    if (newRowIndex < 0) return;
    DataGridViewRow row = dataGridView1.Rows[newRowIndex];
    if (row.Cells["gridName"].Value == null)
        row.Cells["gridName"].Value = "New key";
    if (row.Cells["gridType"].Value == null)
        row.Cells["gridType"].Value = RegistryValueKind.String.ToString();
    if (row.Cells["gridValue"].Value == null)
        row.Cells["gridValue"].Value = "";
}
```

```
// Обробка видалення рядка в DataGridView
private void dataGridView1_UserDeletingRow(object sender, DataGridViewRowCancelEventArgs e)
{
    string name = e.Row.Cells[0].Value?.ToString();

    if (string.IsNullOrEmpty(name)) return;

    if (MessageBox.Show($"Are you sure you want to delete the key '{name}'?", "Confirm Deletion", MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Yes)
    {
        try
        {
            currentKey.DeleteValue(name, false); // Другий параметр 'false' вказує не викидати виняток, якщо значення не існує
            MessageBox.Show("Registry value deleted successfully.", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        catch (SecurityException ex)
        {
            MessageBox.Show($"Permission denied: {ex.Message}\n\nMake sure you are running the application as an administrator and have full control over the registry key.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            RestartAsAdmin();
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Error deleting registry value: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
            }
        }
        else
        {
            e.Cancel = true;
        }
    }

    // Додавання нового ключа реєстру
    private void addToolStripMenuItem_Click(object sender, EventArgs e)
    {
        TreeNode selectedNode = treeView1.SelectedNode;
        if (selectedNode == null) return;

        string name = "New key";

        RegistryKey key = selectedNode.Tag as RegistryKey;
        if (key != null)
        {
            try
            {
                key.CreateSubKey(name);
                MessageBox.Show("Key created successfully.", "Success",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Error creating key: {ex.Message}", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
```

```
}

// Пошук вузла за назвою
private TreeNode findNode(TreeNodeCollection nodes, string part)
{
    foreach (TreeNode node in nodes)
    {
        if (node.Text.Equals(part))
        {
            return node;
        }
    }
    return null;
}

// Обробка зміни вибраного елемента в ComboBox
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    string[] parts = comboBox1.Text.Split('\\');
    if (parts.Length == 0) return;

    TreeNode node = findNode(treeView1.Nodes, parts[0]);
    for (int i = 1; i < parts.Length; i++)
    {
        node = findNode(node.Nodes, parts[i]);
        if (node == null)
            return;
    }
    treeView1.SelectedNode = node;
    node.Expand();
}
```

```
public void SaveRegistryToSQLite(string databasePath)
{
    // Создаем новую базу данных SQLite или подключаемся к существующей
    using (SqliteConnection connection = new SqliteConnection($"Data Source={databasePath};Version=3;"))
    {
        connection.Open();

        // Создаем таблицу для хранения данных реестра
        using (SqliteCommand command = new SqliteCommand(
            "CREATE TABLE IF NOT EXISTS Registry (Path TEXT, Name TEXT, Type TEXT, Value TEXT)", connection))
        {
            command.ExecuteNonQuery();
        }

        // Рекурсивно сохраняем реестр
        SaveRegistryKey(Registry.LocalMachine, "", connection);
        SaveRegistryKey(Registry.CurrentUser, "", connection);
    }
}

private void SaveRegistryKey(RegistryKey key, string path, SqliteConnection connection)
{
    foreach (string valueName in key.GetValueNames())
    {
        object value = key.GetValue(valueName);
        RegistryValueKind valueKind = key.GetValueKind(valueName);
```

```
            string sqlInsert = "INSERT INTO Registry (Path, Name, Type, Value) VALUES (@Path, @Name, @Type, @Value)";
            using (SqliteCommand command = new SqliteCommand(sqlInsert, connection))
            {
                command.Parameters.AddWithValue("@Path", path);
                command.Parameters.AddWithValue("@Name", valueName);
                command.Parameters.AddWithValue("@Type", valueKind.ToString());
                command.Parameters.AddWithValue("@Value", value.ToString());
                command.ExecuteNonQuery();
            }
        }

        foreach (string subKeyName in key.GetSubKeyNames())
        {
            using (RegistryKey subKey = key.OpenSubKey(subKeyName))
            {
                if (subKey != null)
                {
                    SaveRegistryKey(subKey, path + "\\" + subKeyName, connection);
                }
            }
        }
    }

    private void refreshToolStripMenuItem_Click(object sender, EventArgs e)
    {
        TreeNode selectedNode = treeView1.SelectedNode;
        if (selectedNode == null)
        {
```

```
            MessageBox.Show("No key selected for refresh.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }

        OpenFileDialog openFileDialog = new OpenFileDialog
        {
            Filter = "Registry Files (*.reg)|*.reg|SQL (*.sql)|*.sql|All Files (*.*)|*.*",
            Title = "Refresh Registry"
        };

        if (openFileDialog.ShowDialog() == DialogResult.OK)
        {
            try
            {
                switch (openFileDialog.FilterIndex)
                {
                    case 2:
                        LoadRegistryFromSQLite(openFileDialog.FileName);
                        break;
                    default:
                        LoadRegistryFromReg(selectedNode, openFileDialog.FileName);
                        break;
                }
                MessageBox.Show("Key refreshed successfully.", "Success",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Error refresh up key: {ex.Message}", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
        }
    }
}

private RegistryKey GetRegistryKey(string path)
{
    string[] parts = path.Split('\\');
    RegistryKey key = null;

    switch (parts[0])
    {
        case "HKEY_LOCAL_MACHINE":
            key = Registry.LocalMachine;
            break;
        case "HKEY_CURRENT_USER":
            key = Registry.CurrentUser;
            break;
        // Добавьте другие корневые ключи по необходимости
        default:
            return null;
    }

    for (int i = 1; i < parts.Length; i++)
    {
        key = key.OpenSubKey(parts[i], true);
        if (key == null) return null;
    }

    return key;
}
```

```
        private void LoadRegistryFromSQLite(string fileName)
        {
            using (SqliteConnection connection = new SqliteConnection($"Data Source={fileName};Version=3;"))
            {
                connection.Open();
                string sql = "SELECT Path, Name, Type, Value FROM Registry";
                using (SqliteCommand command = new SqliteCommand(sql, connection))
                {
                    using (SqliteDataReader reader = command.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            string path = reader["Path"].ToString();
                            string name = reader["Name"].ToString();
                            string type = reader["Type"].ToString();
                            string value = reader["Value"].ToString();

                            RegistryKey key = GetRegistryKey(path);
                            if (key != null)
                            {
                                RegistryValueKind valueKind = (RegistryValueKind)Enum.Parse(typeof(RegistryValueKind), type);
                                key.SetValue(name, ConvertToRegistryValue(value, valueKind), valueKind);
                            }
                        }
                    }
                }
            }
            // Обновляем отображение
```

```
            LoadRegistryValues(treeView1.SelectedNode);
        }
        private void LoadRegistryFromReg(TreeNode selectedNode, string fileName)
        {
            string content = File.ReadAllText(fileName);
            string[] lines = content.Split(new[] { '\r', '\n' },
StringSplitOptions.RemoveEmptyEntries);

            string currentPath = "";
            foreach (string line in lines)
            {
                if (line.StartsWith("[") && line.EndsWith("]"))
                {
                    currentPath = line.Substring(1, line.Length - 2);
                }
                else if (!string.IsNullOrWhiteSpace(line) && !line.StartsWith(";"))
                {
                    int equalIndex = line.IndexOf('=');
                    if (equalIndex != -1)
                    {
                        string name = line.Substring(0, equalIndex).Trim("");
                        string value = line.Substring(equalIndex + 1).Trim();

                        RegistryKey key = GetRegistryKey(currentPath);
                        if (key != null)
                        {
                            if (value.StartsWith("dword:"))
                            {
                                int intValue = Convert.ToInt32(value.Substring(6), 16);
                                key.SetValue(name, intValue, RegistryValueKind.DWord);
                            }
```

```
                    else if (value.StartsWith("hex:"))
                    {
                        byte[] byteValue = value.Substring(4).Split(',').Select(s => Convert.ToByte(s, 16)).ToArray();
                        key.SetValue(name, byteValue, RegistryValueKind.Binary);
                    }
                    else
                    {
                        key.SetValue(name, value.Trim(""), RegistryValueKind.String);
                    }
                }
            }
        }
        // Обновляем отображение
        LoadRegistryValues(selectedNode);
    }
  }
}
```

Form1.Designer.cs:

```
namespace RegistryEditor
{
  partial class Form1
  {
    /// <summary>
    /// Required designer variable.
    /// </summary>
    private System.ComponentModel.IContainer components = null;

    /// <summary>
```

```
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.treeView1 = new System.Windows.Forms.TreeView();
            this.contextMenuStrip1 = new System.Windows.Forms.ContextMenuStrip(this.components);
            this.addToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.toolStripMenuItem1 = new System.Windows.Forms.ToolStripMenuItem();
            this.toolStripMenuItem2 = new System.Windows.Forms.ToolStripMenuItem();
```

```
        this.dataGridView1 = new System.Windows.Forms.DataGridView();
        this.gridName = new
System.Windows.Forms.DataGridViewTextBoxColumn();
        this.gridType = new
System.Windows.Forms.DataGridViewComboBoxColumn();
        this.gridValue = new
System.Windows.Forms.DataGridViewTextBoxColumn();
        this.comboBox1 = new System.Windows.Forms.ComboBox();
        this.refreshToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
        this.contextMenuStrip1.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).BeginInit();
        this.SuspendLayout();
        //
        // treeView1
        //
        this.treeView1.ContextMenuStrip = this.contextMenuStrip1;
        this.treeView1.Dock = System.Windows.Forms.DockStyle.Left;
        this.treeView1.Location = new System.Drawing.Point(0, 0);
        this.treeView1.Name = "treeView1";
        this.treeView1.Size = new System.Drawing.Size(304, 450);
        this.treeView1.TabIndex = 0;
        this.treeView1.BeforeExpand += new
System.Windows.Forms.TreeViewCancelEventHandler(this.treeView1_BeforeExpan
d);
        this.treeView1.AfterSelect += new
System.Windows.Forms.TreeViewEventHandler(this.treeView1_AfterSelect);
        this.treeView1.NodeMouseClick += new
System.Windows.Forms.TreeNodeMouseClickEventHandler(this.treeView1_NodeM
ouseClick);
```

```
            //
            // contextMenuStrip1
            //
            this.contextMenuStrip1.Items.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.addToolStripMenuItem,
            this.toolStripMenuItem1,
            this.toolStripMenuItem2,
            this.refreshToolStripMenuItem});
            this.contextMenuStrip1.Name = "contextMenuStrip1";
            this.contextMenuStrip1.Size = new System.Drawing.Size(181, 114);
            //
            // addToolStripMenuItem
            //
            this.addToolStripMenuItem.Name = "addToolStripMenuItem";
            this.addToolStripMenuItem.Size = new System.Drawing.Size(180, 22);
            this.addToolStripMenuItem.Text = "Add";
            this.addToolStripMenuItem.Click                    +=                    new
System.EventHandler(this.addToolStripMenuItem_Click);
            //
            // toolStripMenuItem1
            //
            this.toolStripMenuItem1.Name = "toolStripMenuItem1";
            this.toolStripMenuItem1.Size = new System.Drawing.Size(180, 22);
            this.toolStripMenuItem1.Text = "Delete";
            this.toolStripMenuItem1.Click                    +=                    new
System.EventHandler(this.toolStripMenuItem1_Click);
            //
            // toolStripMenuItem2
            //
            this.toolStripMenuItem2.Name = "toolStripMenuItem2";
```

```
        this.toolStripMenuItem2.Size = new System.Drawing.Size(180, 22);
        this.toolStripMenuItem2.Text = "Backup";
        this.toolStripMenuItem2.Click += new
System.EventHandler(this.toolStripMenuItem2_Click);
        //
        // dataGridView1
        //
        this.dataGridView1.AllowUserToOrderColumns = true;
        this.dataGridView1.AutoSizeColumnsMode =
System.Windows.Forms.DataGridViewAutoSizeColumnsMode.Fill;
        this.dataGridView1.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
        this.dataGridView1.Columns.AddRange(new
System.Windows.Forms.DataGridViewColumn[] {
        this.gridName,
        this.gridType,
        this.gridValue});
        this.dataGridView1.Dock = System.Windows.Forms.DockStyle.Fill;
        this.dataGridView1.Location = new System.Drawing.Point(304, 0);
        this.dataGridView1.Name = "dataGridView1";
        this.dataGridView1.Size = new System.Drawing.Size(496, 450);
        this.dataGridView1.TabIndex = 4;
        this.dataGridView1.CellValueChanged += new
System.Windows.Forms.DataGridViewCellEventHandler(this.dataGridView1_CellV
alueChanged);
        this.dataGridView1.UserAddedRow += new
System.Windows.Forms.DataGridViewRowEventHandler(this.dataGridView1_User
AddedRow);
        this.dataGridView1.UserDeletingRow += new
System.Windows.Forms.DataGridViewRowCancelEventHandler(this.dataGridView1
_UserDeletingRow);
```

```
            //
            // gridName
            //
            this.gridName.HeaderText = "Name";
            this.gridName.Name = "gridName";
            this.gridName.Resizable                                    =
System.Windows.Forms.DataGridViewTriState.True;
            this.gridName.SortMode                                     =
System.Windows.Forms.DataGridViewColumnSortMode.NotSortable;
            //
            // gridType
            //
            this.gridType.HeaderText = "Type";
            this.gridType.Name = "gridType";
            this.gridType.Resizable                                    =
System.Windows.Forms.DataGridViewTriState.True;
            this.gridType.SortMode                                     =
System.Windows.Forms.DataGridViewColumnSortMode.Automatic;
            //
            // gridValue
            //
            this.gridValue.HeaderText = "Value";
            this.gridValue.Name = "gridValue";
            //
            // comboBox1
            //
            this.comboBox1.Dock = System.Windows.Forms.DockStyle.Top;
            this.comboBox1.FormattingEnabled = true;
            this.comboBox1.Location = new System.Drawing.Point(304, 0);
            this.comboBox1.Name = "comboBox1";
            this.comboBox1.Size = new System.Drawing.Size(496, 21);
```

```
            this.comboBox1.TabIndex = 6;
            this.comboBox1.SelectedIndexChanged += new
System.EventHandler(this.comboBox1_SelectedIndexChanged);
            //
            // refreshToolStripMenuItem
            //
            this.refreshToolStripMenuItem.Name = "refreshToolStripMenuItem";
            this.refreshToolStripMenuItem.Size = new System.Drawing.Size(180, 22);
            this.refreshToolStripMenuItem.Text = "Refresh";
            this.refreshToolStripMenuItem.Click += new
System.EventHandler(this.refreshToolStripMenuItem_Click);
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(800, 450);
            this.Controls.Add(this.comboBox1);
            this.Controls.Add(this.dataGridView1);
            this.Controls.Add(this.treeView1);
            this.Name = "Form1";
            this.Text = "ErgonReg";
            this.contextMenuStrip1.ResumeLayout(false);

((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).EndInit();
            this.ResumeLayout(false);

        }

        #endregion
```

```
        private System.Windows.Forms.TreeView treeView1;
        private System.Windows.Forms.DataGridView dataGridView1;
        private System.Windows.Forms.ContextMenuStrip contextMenuStrip1;
        private System.Windows.Forms.ToolStripMenuItem toolStripMenuItem1;
        private System.Windows.Forms.ToolStripMenuItem toolStripMenuItem2;
        private System.Windows.Forms.DataGridViewTextBoxColumn gridName;
        private System.Windows.Forms.DataGridViewComboBoxColumn gridType;
        private System.Windows.Forms.DataGridViewTextBoxColumn gridValue;
        private System.Windows.Forms.ComboBox comboBox1;
        private System.Windows.Forms.ToolStripMenuItem addToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem refreshToolStripMenuItem;
    }
}
```

Додаток Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

Перелік файлів на диску

ПЕРЕЛІК ДОКУМЕНТІВ НА МАГНІТНОМУ НОСІЇ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
програма.zip	Архів. Містить коди програми і скомпільовану програму
Презентація	
презентація.ppt	Презентація кваліфікаційної роботи